GUIDE BOOK

MT-N100 LOGIC TRAINER +
ロジック/マイコン トレーナー

ガイドブック ロジック基礎編

Sunhayato

Empower your creativity
78K
Embedded Controller

安全上のご注意

このたびは、弊社製品をご使用いただき、誠にありがとうございます。本項では、誤った取り扱いによる事故を未然に防ぐための安全上の注意事項を説明しています。弊社製品をご使用になる前に必ずお読みください。

⚠警告	この表記を無視して誤った取り扱いをすると、死亡や重傷など、人体への重大な障害をもたらす恐れのある内容について示しています。
⚠注意	この表記を無視して誤った取り扱いをすると、軽傷または中程度の障害をもたらす恐れのある内容について示しています。また、本品や本品に接続している機器に損傷を与える可能性がある事項についても示しています。

⚠警告

水場禁止	▶水分の多いところ、水がかかる場所では、本製品は使用しないで下さい 風呂場や台所など水分の多いところ、水がかかる場所では、本製品は使用しないで下さい。 火災、感電、故障の原因となります。
禁止	▶医療、軍事、航空宇宙、列車、運送、原子力などの制御設備へは使用しないでください 医療機器、軍事機器、航空宇宙機器、運送、原子力などの制御設備などの人命に関わるシステムへの使用は意図しておりません。

⚠注意

分解禁止	▶分解・改造しないでください 分解、改造しないでください。怪我、感電、故障の原因となります。本製品の分解、改造による怪我や事故について、当社は責任を負いかねます。
接触禁止	▶濡れた手での操作は避けてください 濡れた手で電源ケーブル・プラグを抜き差ししないでください。また、製品に触れないでください。感電の原因となることがあります。

⚠注意

禁止	▶以下のような場所では使用しないでください 本製品を以下のような場所で使用すると、動作不良、故障の原因となります。 ・振動や衝撃が加わる場所 ・直射日光のあたる場所 ・湿気やホコリが多い場所 ・温度差の激しい場所 ・熱を発生するもの（暖房器具など）の近く ・強い磁力、電波が発生するもの（磁石、ディスプレイ、スピーカー、ラジオ、無線機など）の近く ・湿気の多い場所

 子供注意	▶子供の手の届かない場所に置いてください 本製品に装着されている電子部品など子供が飲み込まないように注意してください。
 活線挿抜禁止	▶通信ケーブルの抜き差しは、電源ＯＦＦにして行ってください 本品への通信ケーブル類は活線挿抜しないでください。ケーブルの抜き差しは、必ず本製品または相手製品の電源がＯＦＦ状態にて行ってください。故障の原因になることがあります。
 安全設計	▶安全設計をしてください 本製品を、高度な信頼性を必要とするシステムに使用する場合は、冗長設計、誤動作防止設計など充分な安全設計を必ず行ってください。本製品の故障、傷害により生じるいかなる損害、事故について当社は責任を負いかねます。
 保管注意	▶長期間使用しない場合の保管について 長期間使用しない場合は、帯電防止袋などに入れ、ホコリなどが入らないようにしてください。ホコリが入ると接触不良などの原因になります。
 ホコリ注意	▶製品の清掃について 製品にホコリなどが付着すると放熱特性が低下し、故障の原因になりますので、下記の「▶お手入れについて」に従って清掃してください。
 薬品注意	▶お手入れについて ホコリが付着した場合はサンハヤト製ジェットブローなどのガススプレーで吹き飛ばしてください。
 使用注意	▶故障、破損時の処理について 本製品が故障もしくは破損した場合は、速やかに使用を中止して販売店などに修理依頼してください。そのまま使用しますと火災、感電、怪我の原因になるおそれがあります。
 廃棄注意	▶本製品の廃棄について 本製品の廃棄は、各自治体の廃棄ルールに従ってください。詳しくは各自治体にお問い合わせください。

本資料についてのご注意

本資料について

- 本資料は、電子工作や電子回路、パーソナルコンピュータの操作について一般的な知識をお持ちの方を対象にしています。
- 本資料を元に操作するには、株式会社 NEC エレクトロニクス社製マイコンについての知識や開発環境などが必要です。
- Microsoft®、Windows® は米国 Microsoft 社の米国およびその他の国における登録商標です。
- その他、記載されている製品名は各社の商標または登録商標です。

本資料のご利用にあたって

- この取扱説明書に掲載している内容は、お客様が用途に応じた適切な製品をご購入頂くことを目的としています。その使用により当社及び第三者の知的財産権その他の権利に対する保証、又は実施権の許諾を意味するものではありません。また、権利の侵害に関して当社は責任を負いません。
- 本資料に記載した情報を流用する場合は、お客様のシステム全体で充分評価し適用可能かご判断願います。当社では適用可能判断についての責任は負いません。
- 本資料に記載してある内容は、一般的な電子機器（学習教材、事務機器、計測機器、パーソナル機器、コンピュータ機器など）に使用されることを目的としています。高い品質や信頼性が要求され、故障や誤作動が直接人命を脅かしたり人体に危害を及ぼす恐れのある、医療、軍事、航空宇宙、原子力制御、運輸、移動体、各種安全装置などの機器への使用は意図も保証もしておりません。
- この取扱説明書の一部、又は全部を当社の承諾なしで、いかなる形でも転載又は複製されることは堅くお断りします。
- 全ての情報は本資料発行時点のものであり、当社は予告なしに本資料に記載した内容を変更することがあります。
- この資料の内容は慎重に制作しておりますが、万一記述誤りによってお客様に損害が生じても当社はその責任を負いません。
- 本資料に関してのお問合せ、その他お気付きの点がございましたら、当社までお問合せください。
- 本資料に関する最新の情報はサンハヤト株式会社ホームページ（http://www.sunhayato.co.jp/）に掲載しております。

目次

0. はじめに …… 7

0.1 LOGIC TRIANER+ でのロジック学習 …… 8
0.2 論理回路ってなに？ …… 9
0.2.1 ブール代数 …… 9
0.2.2 計算機への応用からコンピュータへの発展 …… 10

1. 開発チュートリアル …… 11

1.1 LOGIC TRAINER+ の各部名称と開発フロー …… 12
1.2 開発用 PC にインストールするもの …… 14
1.3 インストールの手順 …… 15
1.4 LOGIC TRAINER+ を使ってみよう …… 17
1.5 いろいろ配線してみよう …… 26

2. ロジックでつくる …… 27

2.1 基本のロジック …… 28
Exercise 2.1 ①　基本ロジックの動作を確認しよう …… 29
2.2 組み合わせ回路のつくりかた …… 30
2.3 セレクタ …… 32
Exercise 2.3 ①　マルチプレクサをつくろう …… 33
Exercise 2.3 ②　デマルチプレクサをつくろう …… 34
2.4 エンコーダ、デコーダ …… 36
Exercise 2.4 ①　エンコーダをつくろう …… 37
Exercise 2.4 ②　エンコーダの入力に優先順位をつけよう …… 38
Exercise 2.4 ③　デコーダをつくろう …… 40
2.5 加算器 …… 42
Exercise 2.5 ①　加算器をつくろう …… 44

3. 組み合わせ回路と順序回路でつくる …… 45

3.1 基本の順序回路（フリップフロップ） …… 46
3.1.1 フリップフロップの動作原理 …… 47
3.3.2 フリップフロップの種類 …… 51
■ D（Delay）フリップフロップ …… 51
■ T（Trigger）フリップフロップ …… 51
■ JK フリップフロップ …… 52
Exercise 3.1 ①　フリップフロップの動作を確認しよう …… 53
3.2 レジスタ …… 55
Exercise 3.2 ①　シフトレジスタをつくろう …… 56

3.3 カウンタ …… 57
3.3.1 非同期式カウンタ …… 58
3.3.2 同期式カウンタ …… 59
■ 同期式カウンタの考え方 …… 59
Exercise 3.3. ① 非同期式、同期式カウンタをつくろう …… 60

4. トラブル・シューティング …… 63
4.1 回路が書き込めない …… 64
4.1.1 評価ボードが回路書き込みの設定になっていない …… 64
4.1.2 Applilet EZ PL の COM ポート番号が合っていない …… 64
4.2 仮想 COM ポート番号の確認方法 …… 65

改訂履歴 …… 66

コラム目次

論理回路をハードウェアに実現する方法（1） …… 8
論理回路をハードウェアに実現する方法（2） …… 9
論理回路の設計方法 …… 10
USB ーシリアル変換ドライバについて …… 14
論理式を簡単にする方法 …… 31
2 進数について（1） …… 36
Applilet EZ PL：パネルについて調べる …… 39
Applilet EZ PL：ジャンパパネルの関係を調べる …… 40
Applilet EZ PL：複数のパネルをまとめて移動、コピーする …… 41
2 進数について（2） …… 43
2 進数について（3） …… 57
標準ロジック IC について …… 61

0. はじめに

LOGIC TRAINER+（ロジックトレーナープラス）へようこそ！
LOGIC TRAINER+は、マイコンを使った論理回路の学習キットです。
この章では、LOGIC TRAINER+でのロジック学習についてと、論理回路について説明します。

0.1 LOGIC TRAINER+ でのロジック学習
0.2 論理回路ってなに？

0.1 LOGIC TRIANER+ でのロジック学習

LOGIC TRAINER+ は､8 ビットマイクロコンピュータ 78K0S/KA1+(以下､78K0S/KA1+ マイコンといいます）が搭載されている、ロジック学習とマイコン学習のためのトレーニングキットです。

LOGIC TRAINER+ でロジックの学習をする場合は、論理回路作成のためのソフトウェア「Applilet EZ PL」とあわせて使います。「Appilet EZ PL」で作成した論理回路を､78K0S/KA1+ マイコンが実行できる命令に変換し、78K0S/KA1+ マイコン内部のフラッシュメモリに書込みます。書込んだ回路の動作は、LED やスイッチで確認することができます。

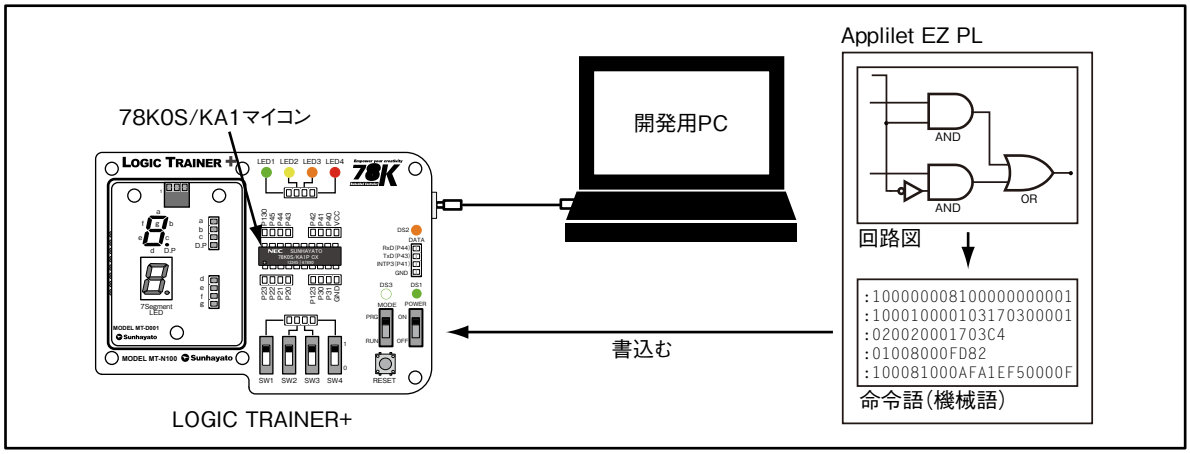


図0-1 LOGIC TRAINER+の論理回路の実現方法（イメージ図）

LOGIC TRAINER+ で開発した論理回路は 78K0S/KA1+ マイコンが命令を実行することで実現するので、論理回路をソフトウェアで実現しているともいえます。一般的な論理回路をハードウェアで実現する方法（詳しくは下コラムを参照してください）とは実行速度や消費電力などが異なりますが、ロジックを学習する上ではどちらも変わりありません。

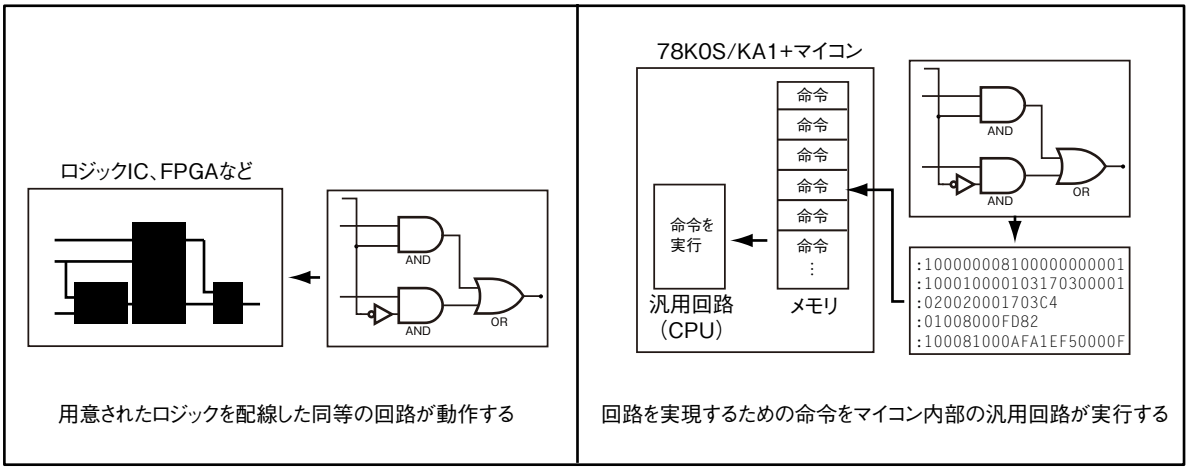


図0-2 一般的なハードウェアによる実現と、LOGIC TRAINER+での実現（イメージ図）

論理回路をハードウェアに実現する方法（1）

一般に個人が学習や開発目的で論理回路をハードウェアで実現するには、ロジック IC（Integrated Circuit）を組み合わせる、PC などを使ってプログラムできる FPGA（Field Programmable Gate Array）という IC を使うなどの方法があります。

これらの方法は、IC の主な素材であるシリコンにあらかじめ作りこまれたロジックを配線する（FPGA では、ロジック間の配線をメモリにプログラムしています）ことで、回路を実現しています。

0.2 論理回路ってなに？

電圧の高い、低いを「1」「0」とみなし、この 2 つの数値のみを扱う論理回路は、ディジタル回路とも呼ばれます。論理回路はどのようないきさつで生まれたのでしょうか。

0.2.1 ブール代数

論理回路とは、ブール代数の考え方に基づいて設計された回路のことです。ブール代数とは、19 世紀の数学者・哲学者のジョージ・ブール（イギリス 1815 ～ 1864）が考案した、論理学を数学的に解く考え方です。論理学では、あるテーマ（命題といいます）が正しいか正しくないかを論じますが、ブール代数では命題を A、B などの記号に置き換え、正しい（真）を 1 に、正しくない（偽）を 0 として、以下の 3 つの論理演算を使います。

表_基本の3つの論理演算

名称	論理記号	論理	演算規則
論理和	＋（OR）	どちらかが 1 なら 1	1+1=1　1+0=1 0+1=1　0+0=0
論理積	・（AND）	どちらも 1 なら 1	1･1=1　1･0=0 0･1=0　0･0=0
論理否定	$\overline{A}$（NOT） （記号の上にーを付ける）	1 なら 0、0 なら 1	$\overline{1}$=0　$\overline{0}$=1

また、ブール代数には以下の定理があります。論理式を簡単にするときに使用します。

表_ブール代数の諸定理

名称	公式	名称	公式
公理	1+A=1 0･A=0	交換の法則	A+B=B+A A･B=B･A
項等	0+A=A 1･A=A	結合の法則	A+(B+C)=(A+B)+C A･(B･C)=(A･B)･C
同一の法則	A+A=A A･A=A	分配の法則	A･(B+C)=A･B+A･C A+B･C=(A+B)･(A+C)
補元の法則	A+$\overline{A}$=1 A･$\overline{A}$=0	吸収の法則	A･(A+B)=A, A+A･B=A A+$\overline{A}$･B=A+B, $\overline{A}$+A･B=$\overline{A}$+B
復元の法則※	$\overline{\overline{A}}$=A	ド・モルガンの法則	$\overline{A+B}=\overline{A}\cdot\overline{B}$ $\overline{A\cdot B}=\overline{A}+\overline{B}$

※ $\overline{\overline{A}}$ は $\overline{A}$ の否定を表します

論理回路をハードウェアに実現する方法（2）

LOGIC TRAINER+ に搭載している 78K0S/KA1+ マイコンや、PC の CPU も論理回路で設計されていますが、回路の集積度の高い LSI（Large Scale Integrated circuit）という IC が使われています。FPGA では、IC 内にあらかじめつくりこまれたロジックを配線して、所望の回路を実現する方法をとりますが、マイコン、CPU ほどの規模になると最初から回路を実現するロジックだけを IC につくりこむ形になります。この方法はフルカスタム IC と呼ばれ、大規模な設備で大量生産されます。

0.2.2 計算機への応用からコンピュータへの発展

20 世紀に入り、電気回路のスイッチの ON、OFF をブール代数での「1」「0」とし、直列回路で「AND」、並列回路で「OR」の論理を実現できることがわかり、計算機への応用が考えられました。

さらに「1」「0」の 2 つの数字で 2 進数の計算をしたり、様々な情報を表現できることがわかりました。「1」「0」の 2 つの数値を表す、2 つのレベルの電圧を実現する回路は、それまでのアナログ回路に対してディジタル回路とも呼ばれます。

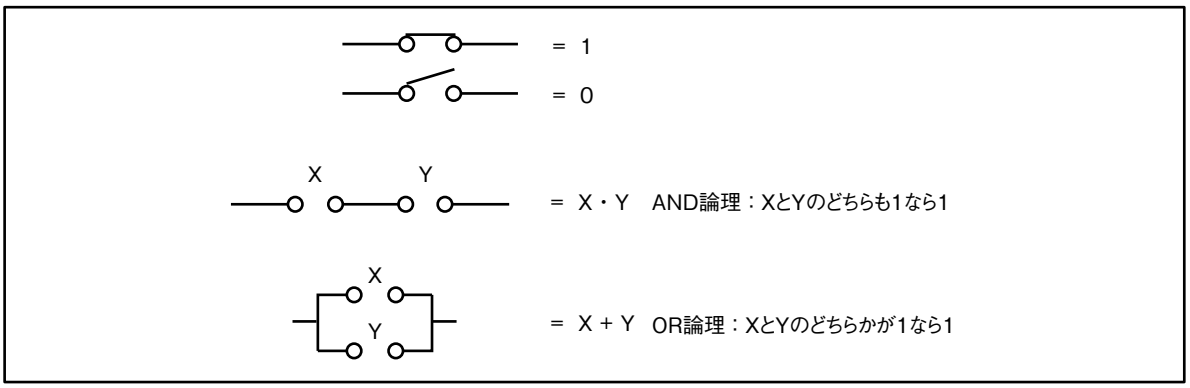


図0-3 ブール代数の論理をスイッチ回路で実現する

ディジタル回路はアナログ回路に比べて、いろいろな利点がありました。「1」「0」を区別するための 2 つの電圧レベルを実現すればよいので、電子部品の特性のばらつきや、温度、ノイズなどの影響をあまり受けません。設計においても、ディジタル回路は与えられる入力値に対して、所望の出力値を実現する論理のみを考えればよく、電気的な要素とは切り離せます。

ブール代数の論理を実現する論理素子はその後、リレースイッチから真空管、パラメトロン、トランジスタへと進歩しました。論理素子が小型化、省電力化するに従い論理回路の集積度も上がり、より多くの処理が出来るコンピュータへと発展しました。

現在はトランジスタをより微細にシリコンに作りこむ IC が登場し、ロジック IC、プログラマブルな FPGA、フルカスタム IC など、回路の規模や生産する数、設備等から選択できます。

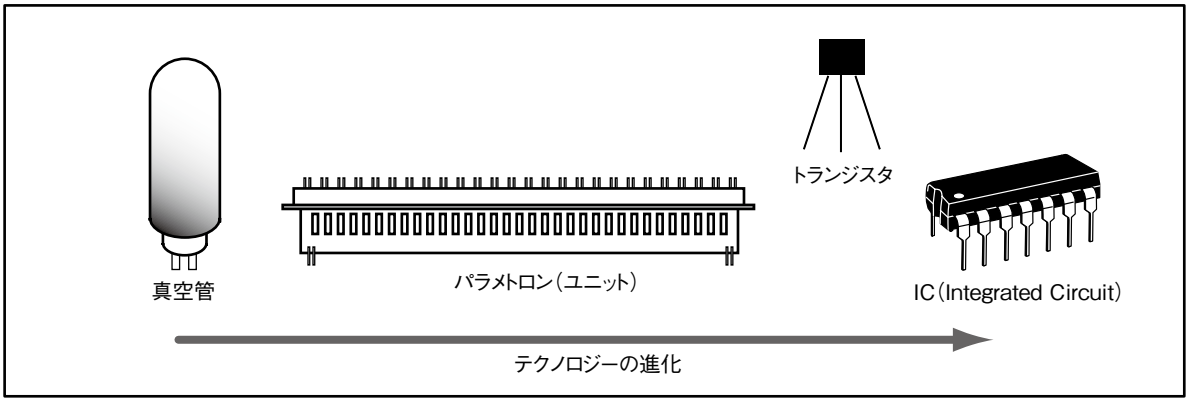


図0-4 論理素子の進化

論理回路の設計方法

論理回路の設計方法も、テクノロジーの発展に伴い変化しています。当初は論理式から論理の簡単化を行い、紙に論理回路図を描いていましたが、CAD（Computer Aided Design）が使われるようになると論理回路図だけ描いて論理の簡単化はコンピュータがしてくれるようになりました。現在では回路図を描く代わりに VHDL などのハードウェア記述言語で動作を記述する方法が多くとられています。

1. 開発チュートリアル

この章では、LOGIC TRAINER+で学習するための準備と、LOGIC TRAINER+でロジックを配置して回路を作成し、動作を確認するまでの手順を説明します。

1.1 各部の名称と開発フロー
1.2 開発用 PC にインストールするもの
1.3 インストールの手順
1.4 LOGIC TRAINER+ を使ってみよう
1.5 いろいろ配線してみよう

1.1 LOGIC TRAINER+ の各部名称と開発フロー

LOGIC TRAINER+ の各部名称を以下に示します。

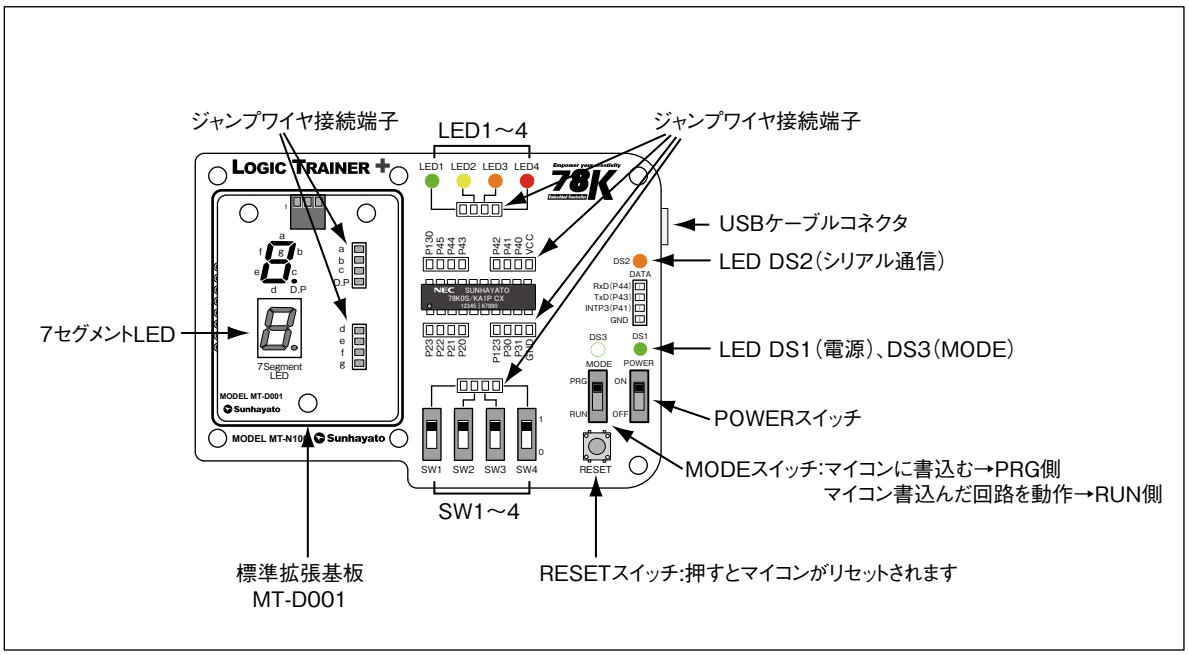


図1-1 LOGIC TRAINER+の各部名称

LOGIC TRAINER+ では、マイコンの端子とスイッチや LED をジャンプワイヤで接続することで、マイコンからの制御を可能にします。ジャンプワイヤは差込口にまっすぐゆっくりと差し込んでください。また、ジャンプワイヤの配線を変更する場合は、必ず LOGIC TRIANER+ の POWER スイッチを OFF 側にして電源を切ってください。

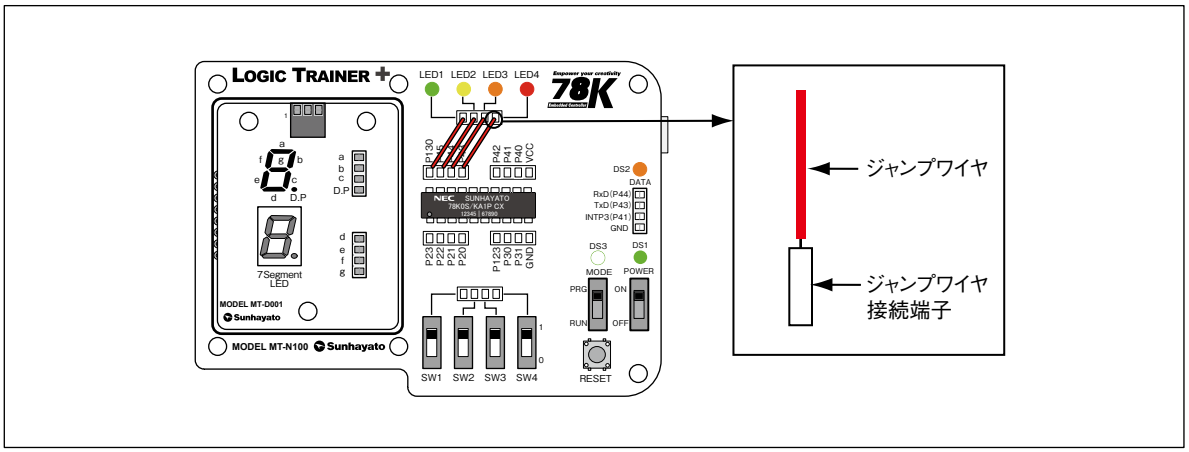


図1-2 ジャンプワイヤによる配線

LOGIC TRAINER+ の開発フローは以下のようになります。
Applilet EZ PL でロジックを配置して回路を作成し、USB ケーブルを介して LOGIC TRAINER+ に書込みます。

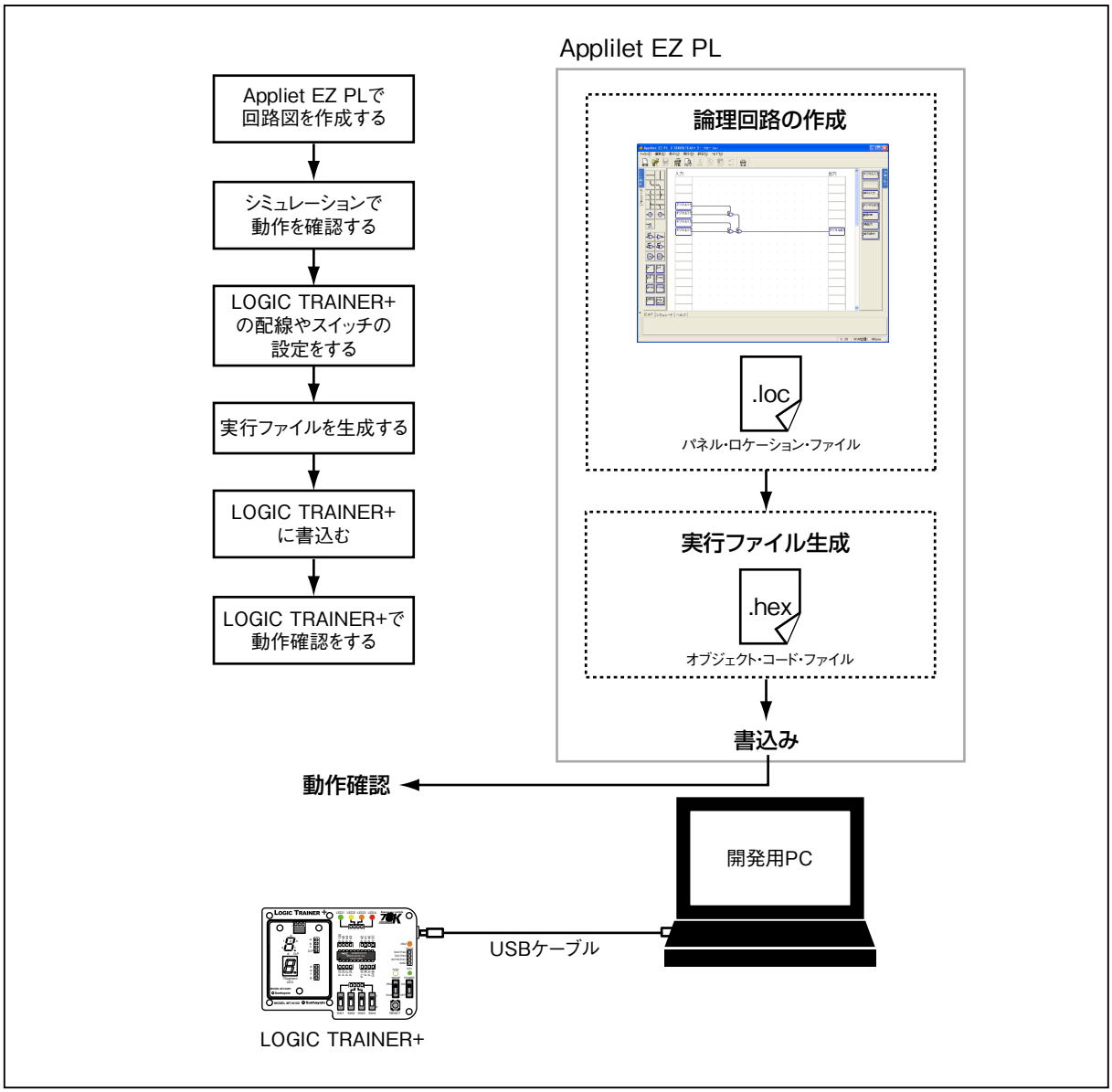


図1-3 LOGIC TRAINER+ 開発フロー

1.2 開発用 PC にインストールするもの

LOGIC TRAINER+ で開発をするために、以下のソフトウェアを開発用 PC にインストールします。

① **USB －シリアル変換ドライバ**

開発用 PC から USB 通信で LOGIC TRAINER+ に書込むために必要なドライバです。

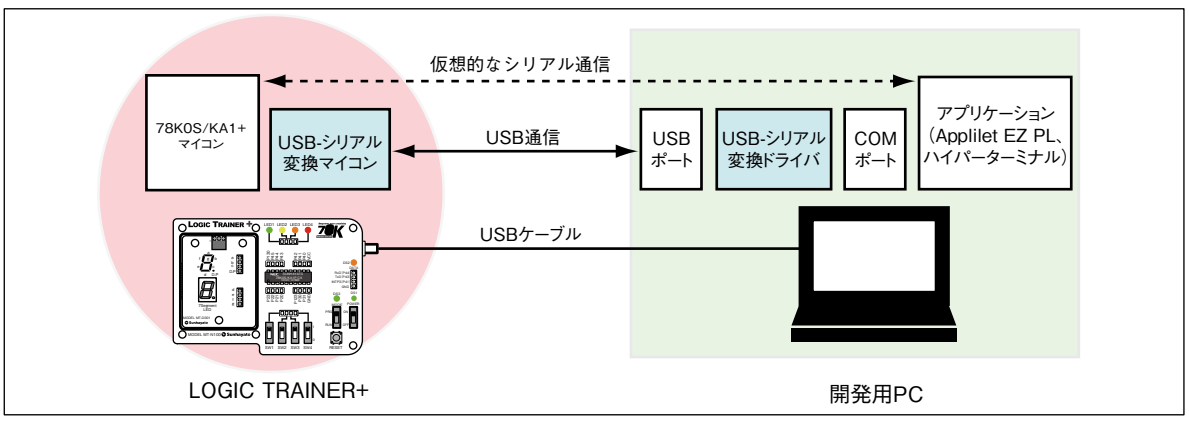
② **Applilet EZ PL**

ロジックを配置して論理回路を作成するソフトウェアです。

USB－シリアル変換ドライバについて

開発用 PC から USB 通信で LOGIC TRAINER+ に書込むために必要なドライバです。LOGIC TRAINER+ には論理回路を実現する 78K0S/KA1+ マイコンの他に、USB －シリアル変換マイコンが搭載されており、この変換マイコンに対応するドライバがインストールされている PC と、USB ケーブルを介して仮想的なシリアル通信をおこなうことができます。

仮想的なシリアル通信とはこの場合、実際は USB ケーブルで接続して USB 通信を行っていますが、ターゲット・マイコンとアプリケーションの間ではあたかもシリアル通信を行っているような通信のしくみをいいます。



1.3 インストールの手順

① USB －シリアル変換ドライバのインストール

開発用 PC に、LOGIC TRAINER+ に付属の CD-ROM を挿入してください。また、開発用 PC と LOGIC TRAINER+ を USB ケーブルで接続してください。（LOGIC TRAINER+ は、プラスチックケースから出してお使いください。）

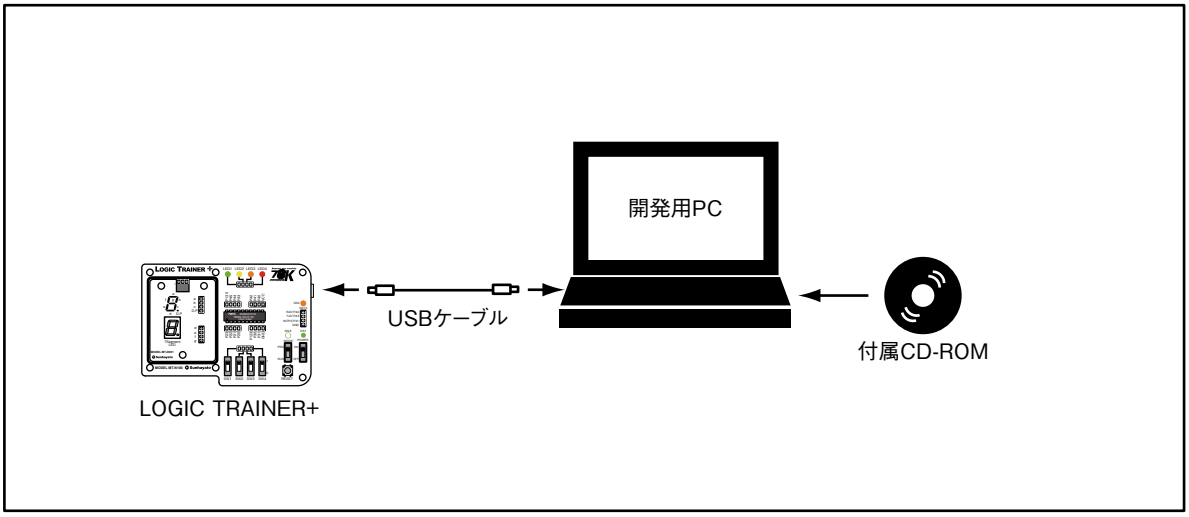


図1-6 LOGIC TRAINER+ のセットアップ

開発用 PC が LOGIC TRAINER+ と接続したことが認識されると、「新しいハードウェアの検索ウィザードの開始」ウィンドウが開きます。以下の手順に従って、CD-ROM より USB －シリアル変換ドライバをインストールしてください。

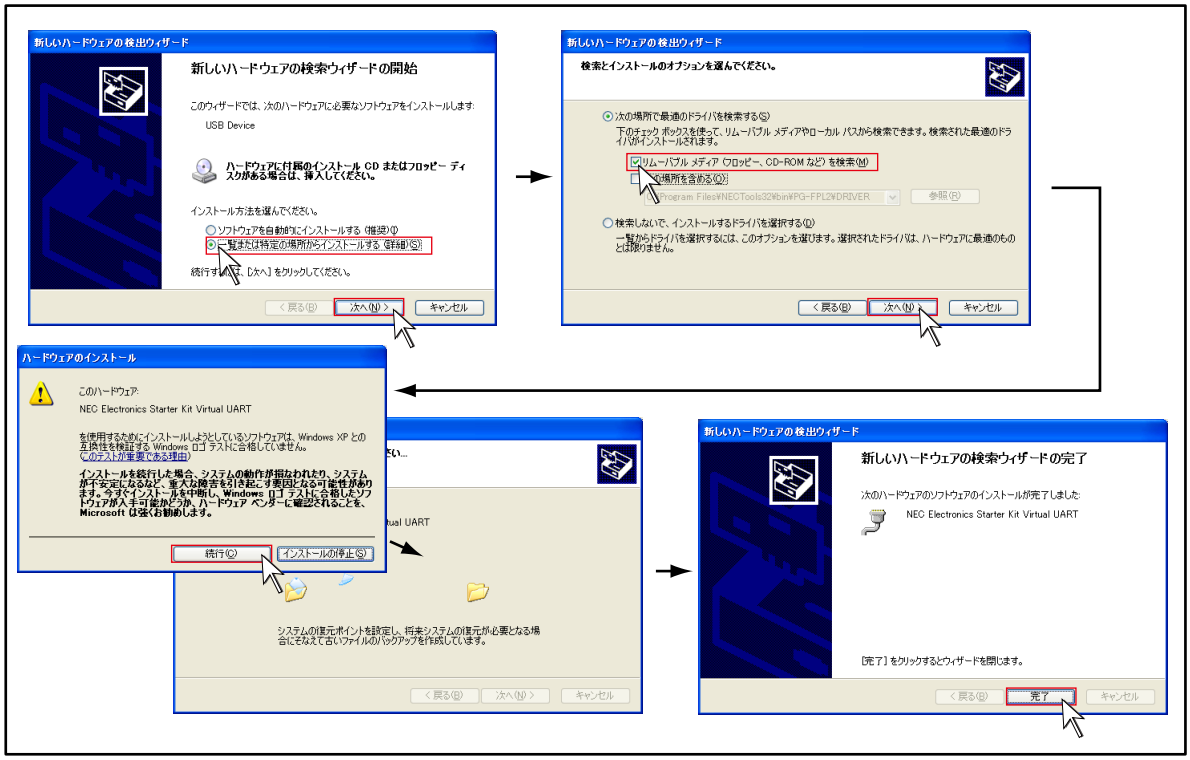


図1-7 USB-シリアル変換ドライバのインストール手順

② Applilet EZ PL のインストール

LOGIC TRAINER+ に付属の CD-ROM を挿入したときに、自動的に開くブラウザよりインストールできます。ブラウザが自動的に開かない場合は、CD-ROM 内の HTML ファイル「LOGIC_TRAINER_plus.html」をダブルクリックして手動で開いてください。

ブラウザの「Applilet EZ PL のインストール」項の「ここをクリックするとインストールが始まります」のリンクをクリックすると、インストールが始まります。インストーラが示す手順に従ってインストールしてください。インストールするフォルダは特に問題がない限り、インストーラのデフォルトの設定のまま変更しないことをお勧めします。

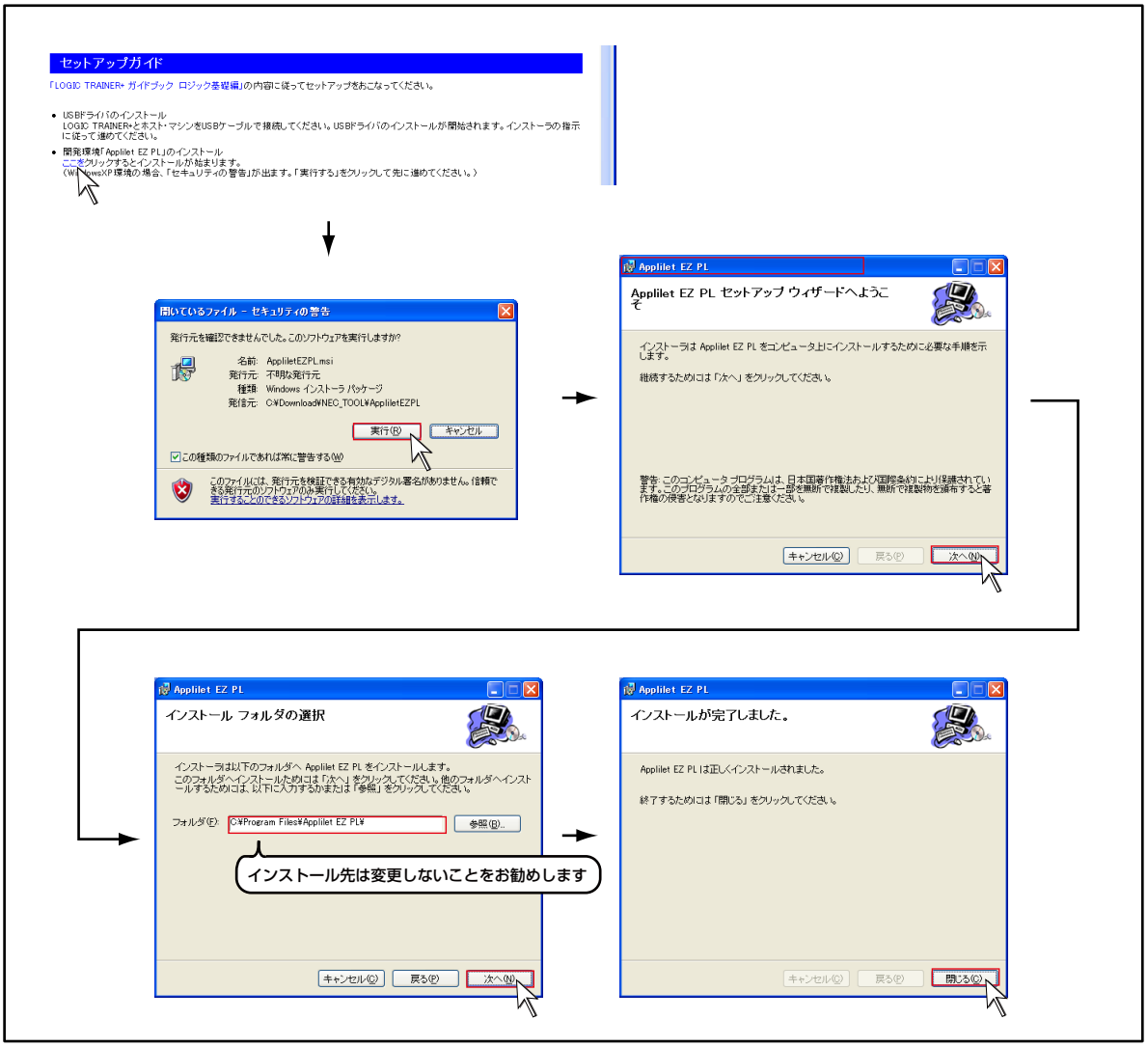
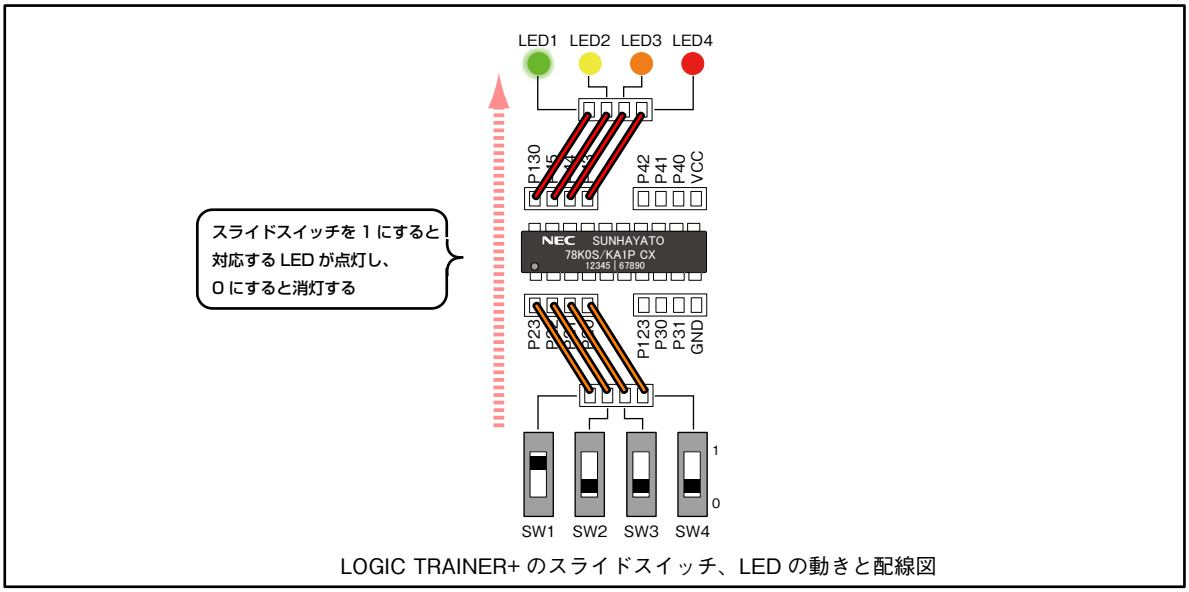


図1-8 Applilet EZ PLのインストール手順

1.4 LOGIC TRAINER+ を使ってみよう

Applilet EZ PL で簡単な回路を作成し、LOGIC TRAINER+ に書込んでみましょう。
LOGIC TRAINER+ の SW1、2、3、4 の状態を LED 1 、2、3、4 に出力する回路をつくります。



LOGIC TRAINER+ のスライドスイッチ、LED の動きと配線図

図1-9 LOGIC TRAINER+のスライドスイッチとLED

① Applilet EZ PL で回路図を作成する

開発用 PC にインストールされた Applilet EZ PL を起動します。Windows のデスクトップのアイコンをダブルクリックしても、「スタート」メニューからでも起動できます。

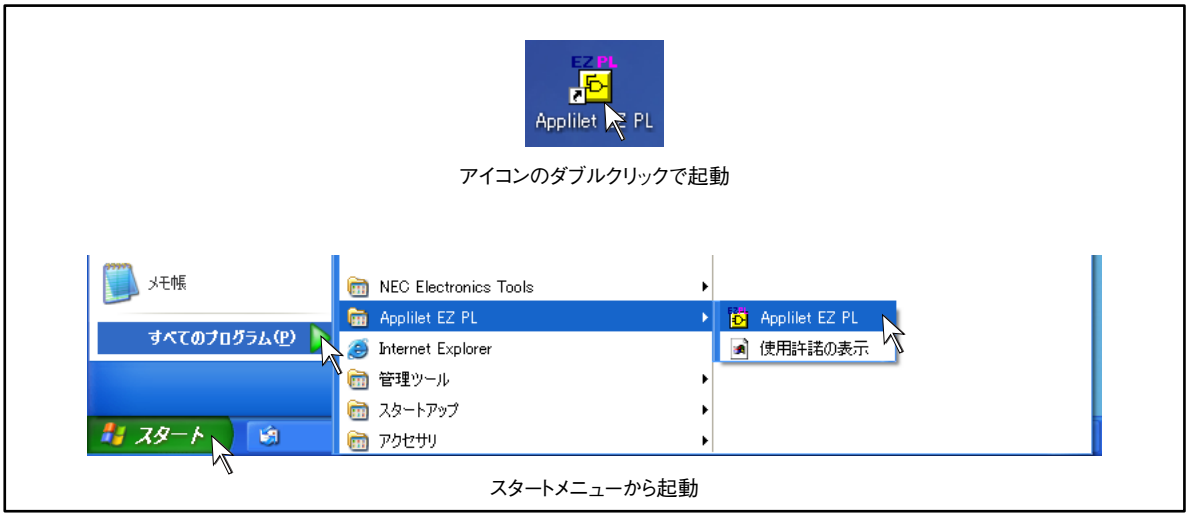


図1-10 Applilet EZ PLの起動

Applilet EZ PL が起動したら、メイン・ウィンドウが開きます。ここに回路図を作成します。
メイン・ウィンドウ右側に入力、出力パネルが用意されています。入力、出力パネルでは、マイコンのどの端子から信号を入出力するかを設定します。これらのパネルをメイン・ウィンドウの入力 / 出力部に配置していきます。このガイドで主に扱う LED やスイッチなどの入出力機器を使うときは、「デジタル入力」、「デジタル出力」パネルを配置します。
メイン・ウィンドウ左側には処理パネルが用意されています。処理パネルは処理部に配置して、入力パネルから入力された信号を接続したり加工したりします。

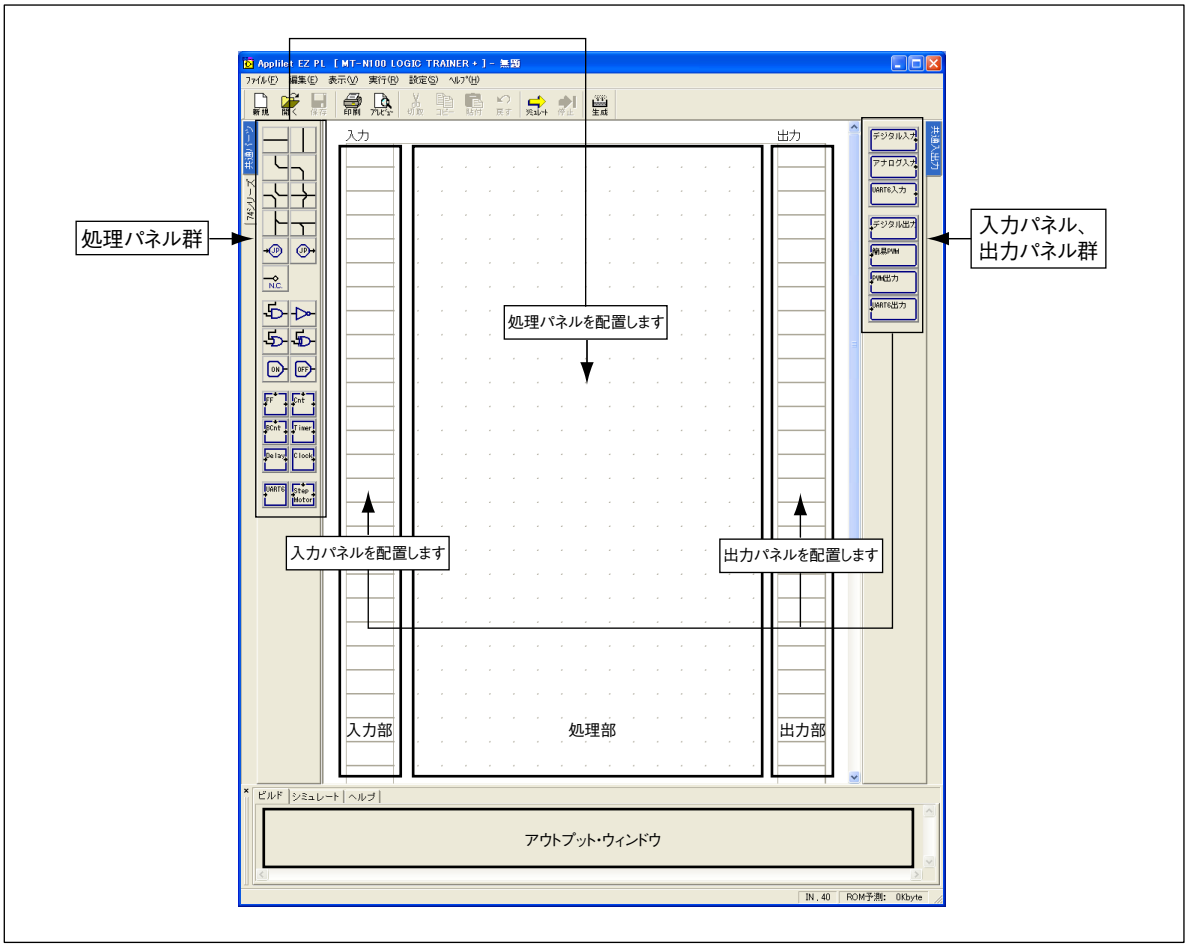


図1-11 Applilet EZ PL：メイン・ウィンドウ

② ターゲット CPU の設定

まず､回路を書込む CPU(ここでは“マイコン”と同じ意味です)の種類を設定します。「設定」メニューの「ターゲット CPU」をクリックしてください。「ターゲットの設定」ボックス内の「LOGIC TRAINER+」を選択して「OK」をクリックしてください。メイン・ウィンドウ上の表示が「LOGIC TIRANER+」となったことを確認してください。この設定で、LOGIC TRAINER+ に搭載されているマイコンに、回路を書込むことができます。

この設定は一度行うと、再度違う設定を行わないかぎり、Applilet EZ PL を再起動しても有効です。

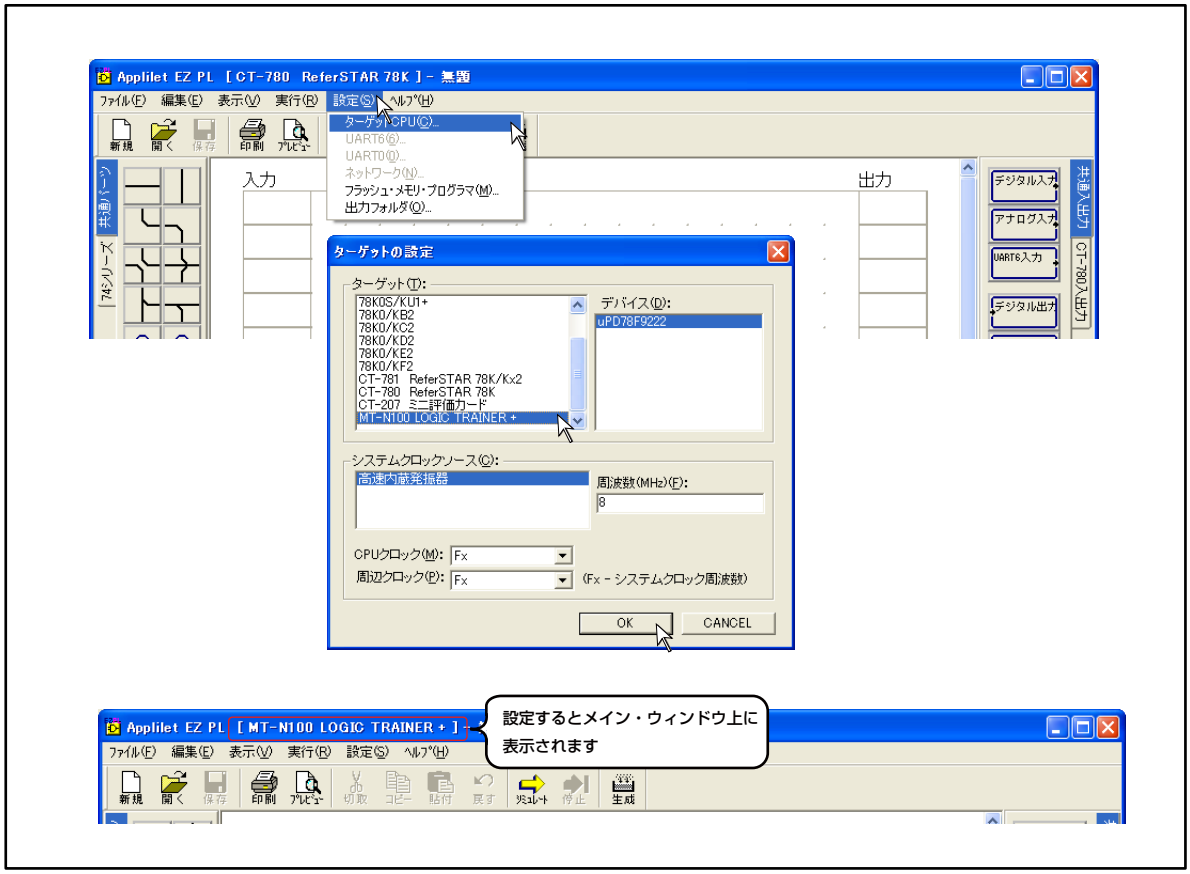


図1-12 Applilet EZ PL：ターゲットCPUの設定

入出力パネルを配置するスライドスイッチ用に４つの「デジタル入力」を、LED 用に４つの「デジタル出力」パネルを配置してみましょう。パネルは、マウス左クリックでそのまま配置したい場所までドラッグすると移動することができます。

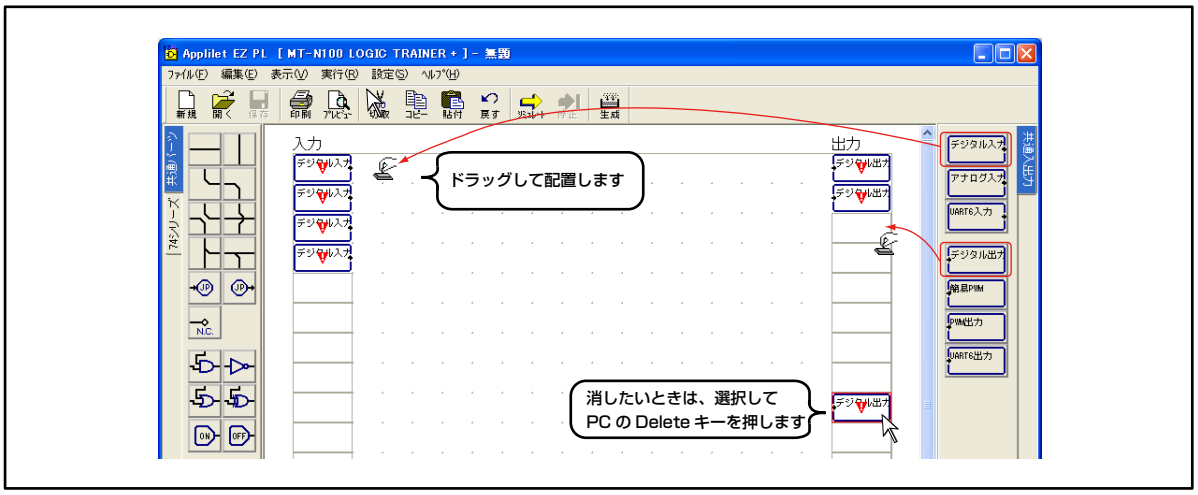


図1-13 Applilet EZ PL：入出力パネルの配置

③ 入出力パネルの接続先を設定する

入出力パネルを配置したら、各入出力パネルにマイコンの端子を設定します。

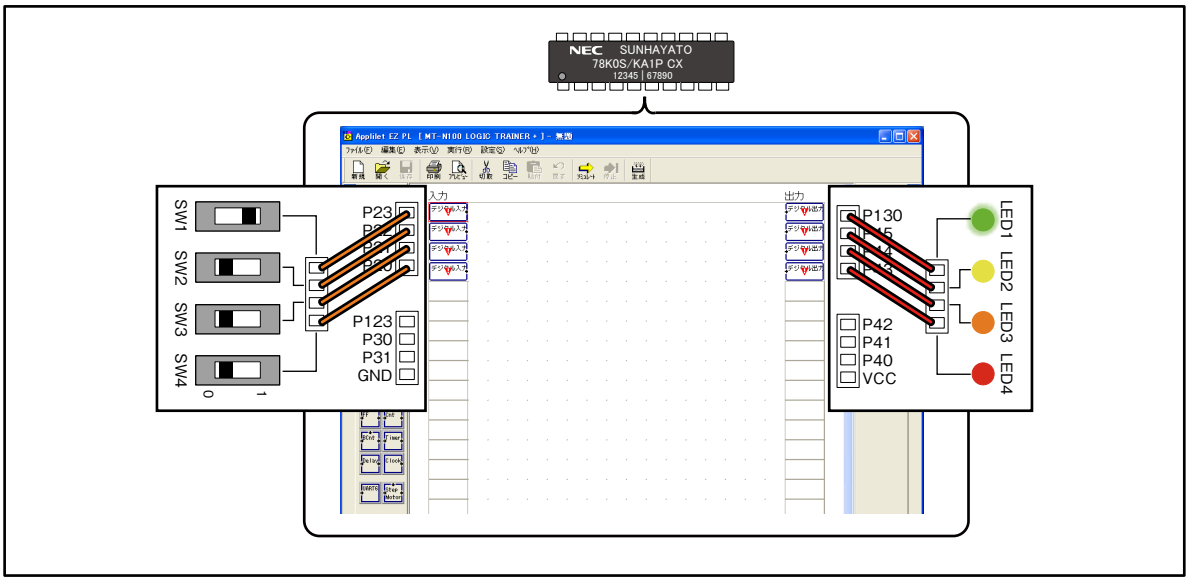


図1-14 Applilet EZ PL：マイコンに接続しているスイッチ、LED（イメージ図）

各入出力パネルをダブルクリックすると、プロパティ・ウィンドウが表示され、マイコンのポートを設定することができます。ここでは､LOGIC TRAINER+ での配線に合わせて以下のように設定することとします。「アクティブ」､「プルアップ抵抗」の設定の変更は必要ありません。

入力パネル		出力パネル	
割り当てポート	表示名称※	割り当てポート	表示名称※
P23	SW1-P23	P130	LED1-P130
P22	SW2-P22	P45	LED2-P45
P21	SW3-P21	P44	LED3-P44
P20	SW4-P20	P43	LED4-P43

※入出力パネルの「表示名称」はわかりやすい名前を付けてください。

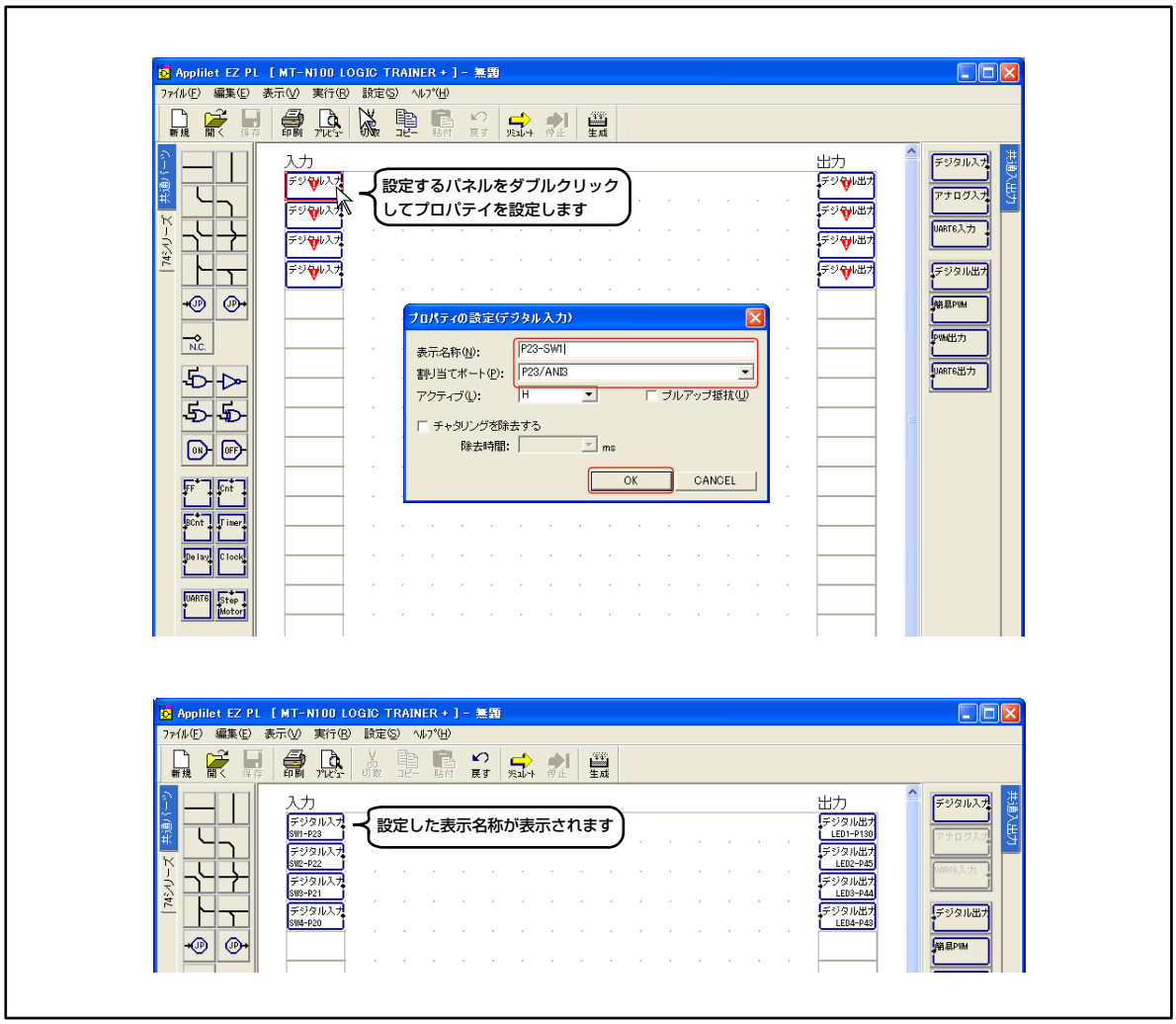


図1-15 Applilet EZ PL：入出力パネルのプロパティ設定

④ 配線をする

入力パネルから入力された信号が、出力パネルへ送られるように配線をします。メイン・ウィンドウ左に用意された配線パネルで、接続するのに都合の良いものを選んで配置していきます。ここでは、以下のように接続することとします。

入力パネル		出力パネル
P23	→	P130
P22	→	P45
P21	→	P44
P20	→	P43

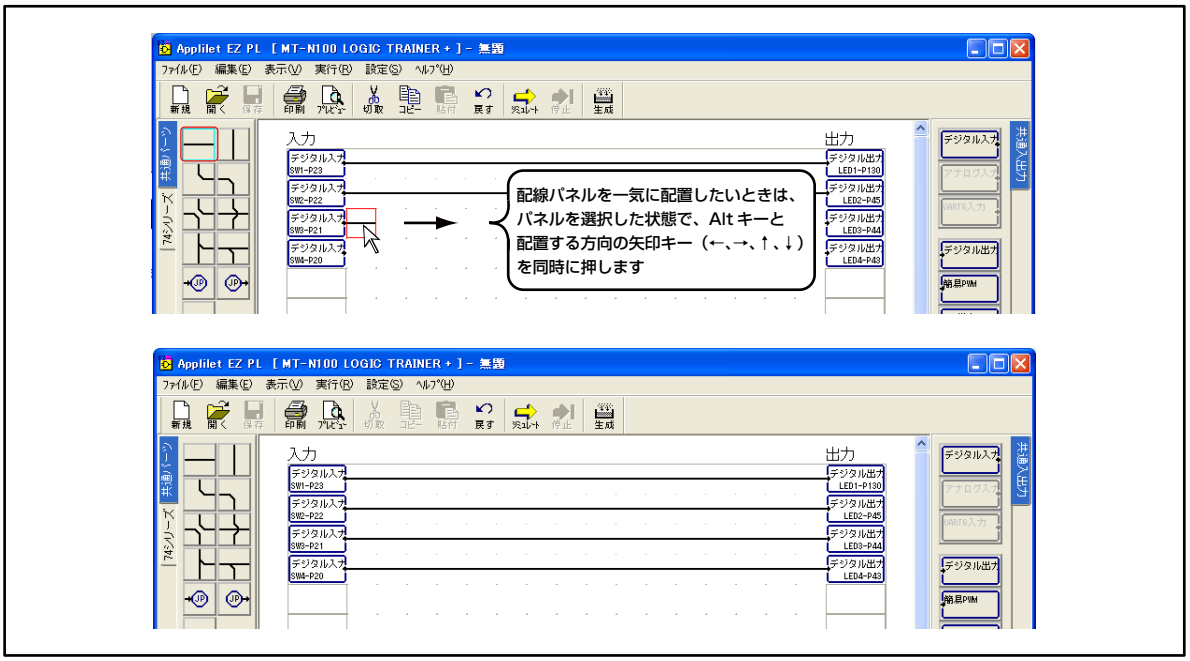


図1-16 Applilet EZ PL：配線

⑤ 回路図を保存する

回路図が完成したら､名前を付けて保存しておきます。「ファイル」メニューの「名前を付けて保存」をクリック、名前を付けて保存してください。回路図ファイル（パネルロケーションファイル）は､「マイ ドキュメント」フォルダ内の「Applilet EZ PL」フォルダ内の「SAVE」フォルダに保存されます。

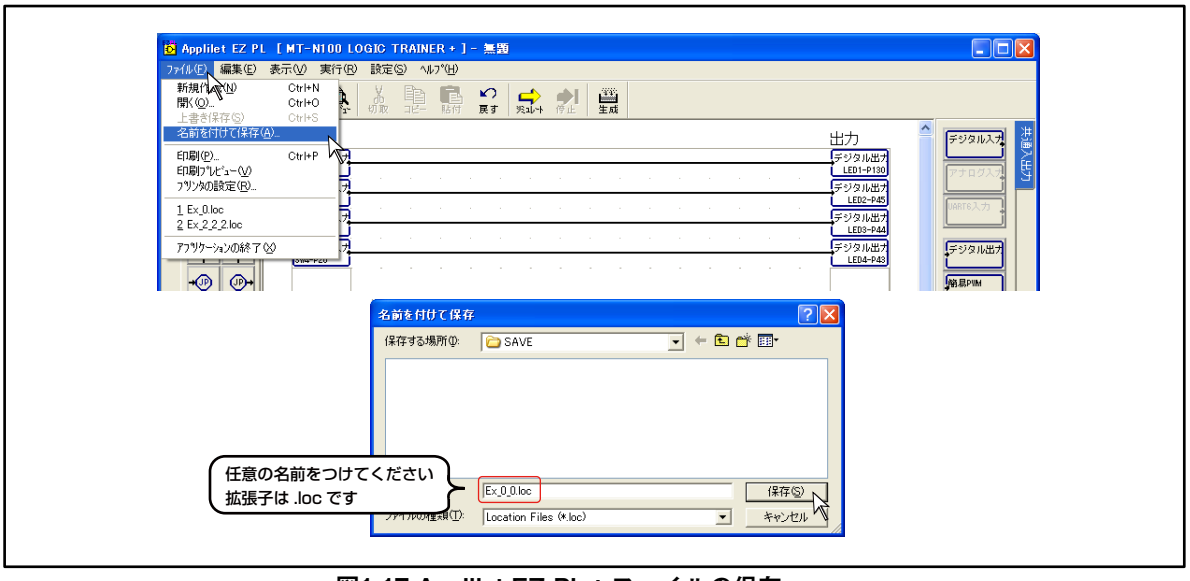


図1-17 Applilet EZ PL：ファイルの保存

⑤ 回路をシミュレートする

Applilet EZ PL には、作成した回路をシミュレート（模擬的に動作させる）できるシミュレーション機能があります。LOGIC TRAINER+ に書込む前に、開発 PC 上で意図どおりに回路が動作するかどうかを試すことができます。

「ファイル」メニューの「シミュレート」か、「シミュレート」ボタンをクリックしてください。メイン・ウィンドウ下にシミュレート結果の波形が表示されます。

入力パネルをクリックすると、入力信号が切り替わり、結果を波形で確認することができます。

シミュレーションを停止するには、「ファイル」メニューの「シミュレート停止」か、「停止」ボタンをクリックしてください。

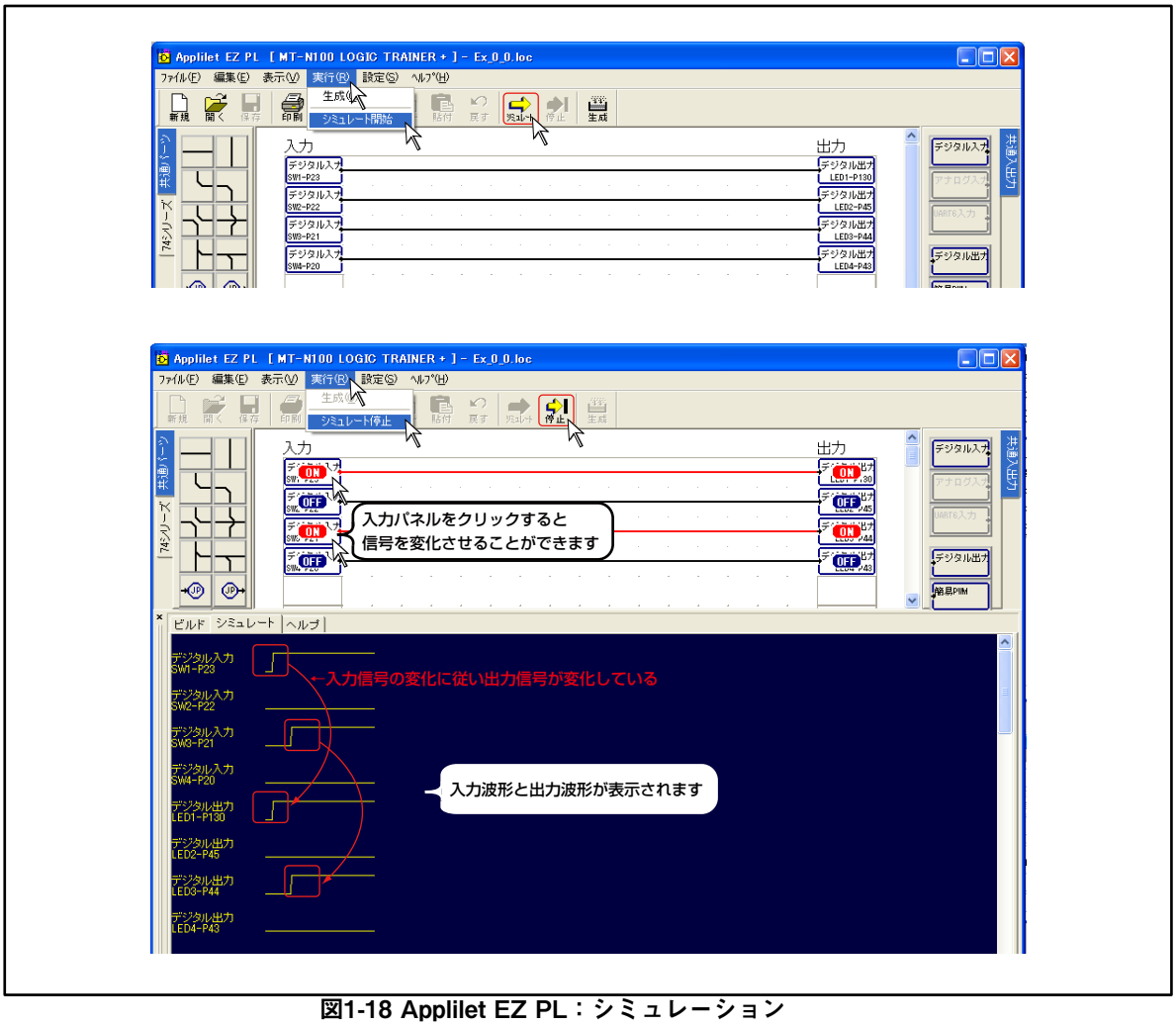


図1-18 Applilet EZ PL：シミュレーション

⑥ LOGIC TRAINER+ の準備をする

回路図を保存し、動作を確認したら実行ファイルを生成しますが、実行ファイル生成→ 78K0S/KA1+ マイコンへの書込みと続けておこなわれますので、まず LOGIC TRAINER+ 本体の準備をしておきます。
LOGIC TRAINER+ の USB コネクタ CN1 に USB ケーブルを接続し、開発用 PC にも接続してください。
以下の手順で LOGIC TRAINER+ の各設定をおこなってください。

（a）ジャンプワイヤーでスイッチや LED を正しく 78K0S/KA1+ マイコンのピンに接続してください。

（b）POWER スイッチを「ON」に設定し、LED DS1 が点灯することを確認してください。

（c）回路図を書込むために MODE スイッチを「PRG」側に設定してください。
LED DS3 が赤色に点灯することを確認してください。

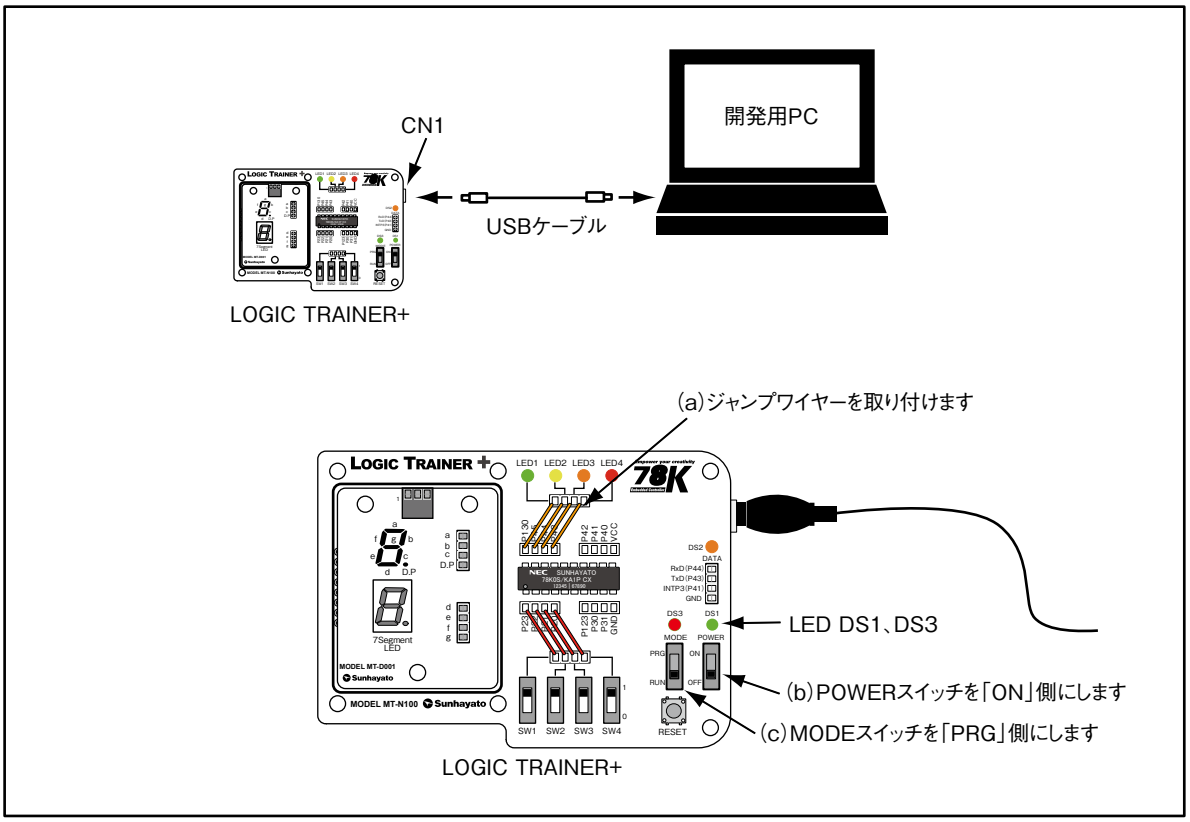


図1-19 LOGIC TRAINER+ へ書込むための準備

注意 780S/KA1+ マイコンへの書込みの最中は、決して LOGIC TRAINER+ の電源や開発用 PC の電源を切らないようにしてください。。

⑦ 回路図から実行ファイルを生成する

Applilet EZ PL で実行ファイルを生成します。「実行」メニューの「生成」か、「生成」ボタン をクリックしてください。生成が成功したら続いて書込みを行うか確認するウィンドウが表示されますので、「OK」をクリックして書込みを行います。

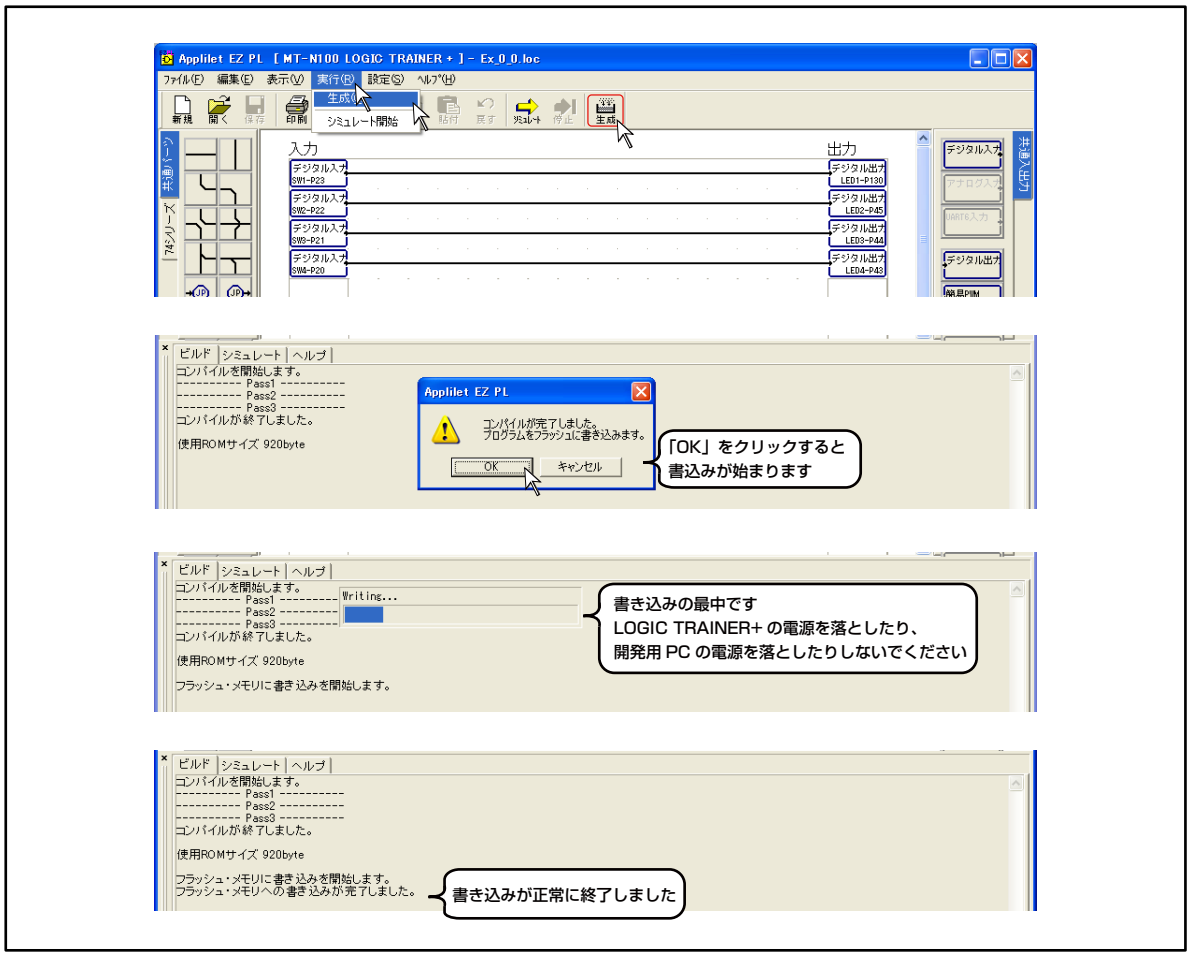


図1-20 Applilet EZ PL：生成と書込み

● 「コンパイル」でエラーとなった場合

もう一度回路図を見直してください。入力パネルと出力パネルが完全に接続されていなかったり、余分な配線パネルがある場合は、コンパイルエラーとなります。

● 「書き込み」でエラーとなった場合

Windows のデバイスマネージャで「Virtual COM ポート」に「!」が表示されていたら、USB が正しく認識されていません。一旦 USB ケーブルを抜き差ししてください。それでも書込めない場合は、USB―シリアル変換ドライバを再インストールしてください。再インストールの手順は、「1.3 インストールの手順」(P14)と同じになります。

Windows のデバイスマネージャで「Virtual COM ポート」に「!」が表示されていない場合は、Applilet EZ PL に正しい仮想 COM ポートが自動検出されていない場合があります。Applilet EZ PL の「設定」メニューの「フラッシュ・メモリ・プログラマ」をクリックして、「COM ポート」の設定を「自動検出」から実際に割り当てられている仮想 COM ポート番号に変更してください。

Windows のデバイスマネージャの表示、現在割り当てられている仮想 COM ポート番号の調べ方については、「4. トラブルシューティング」（P64）を参照してください。

⑧ LOGIC TRAINER+ で動作を確認する

LOGIC TRAINER+ の MODE スイッチを「RUN」側に設定します。LED DS3 が緑色に点灯することを確認してください。

スイッチ SW1 ～ 4 を 1 ⇔ 0 に設定して、LED1 ～ 4 が書き込んだ回路どおりに動作するか確認してください。

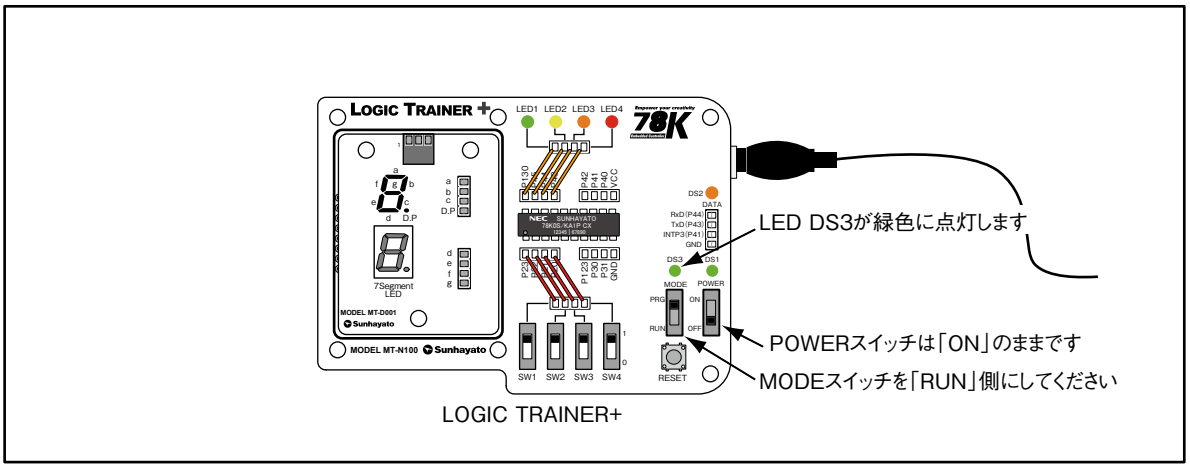


図1-21 LOGIC TRAINER+での動作確認

1.5 いろいろ配線してみよう

LOGIC TRAINER+ に付属の標準拡張基板（MT-D001）にも配線して、7 セグメント LED の各セグメントを、スライドスイッチから制御してみましょう。ジャンプワイヤの接続の変更は、必ず LOGIC TRAINER+ の電源スイッチを OFF に設定し、電源を切ってからおこなってください。使うマイコンのピンや、接続先を入力機器（スライドスイッチ）から出力機器（LED など）へ、あるいはその反対に変更した場合は、Applilet EZ PL で回路図の変更が必要です。

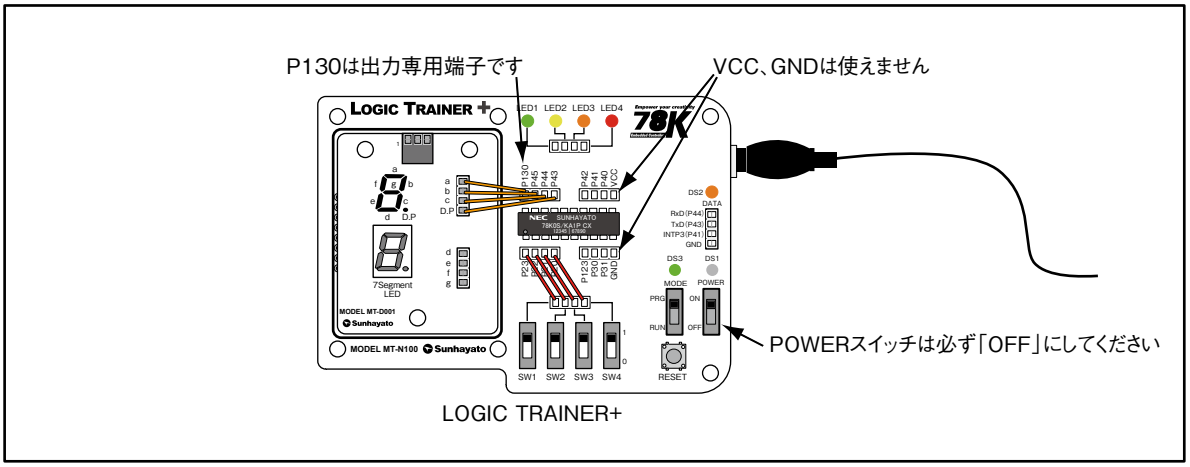


図1-22 LOGIC TRAINER+ いろいろな配線

注意

78OS/KA1+ マイコンの P130 ピンは出力専用端子なのでスイッチ等入力機器には接続できません。
また、VCC は常に 1 を出力する端子、GND 端子は常に 0 を出力する端子です。

2. ロジックでつくる

ロジックを組み合わせた回路を、組み合わせ回路といいます。この章では、基本のロジックと、組み合わせ回路を紹介します。

2.1 基本のロジック
2.2 マルチプレクサ、デマルチプレクサ
2.3 デコーダ、エンコーダ
2.4 加算回路
2.5 減算回路

2.1 基本のロジック

基本の論理を実現する最小単位の回路をロジック、あるいは論理ゲートといいます。デジタル信号は、このゲート（門）回路を通って加工され､出力されます。基本となるロジックは論理和 AND（アンド)､論理積 OR（オア)、論理否定 NOT(ノット)の 3 種類です。論理回路はすべて､この 3 つのロジックから成り立っているともいえます。

論理回路を描く場合は、各ロジックを MIL 記号という記号で表します。また、ロジックの動作を表す表を真理値表といいます。以下に基本のロジックの MIL 記号とその動作を表す真理値表を示します。

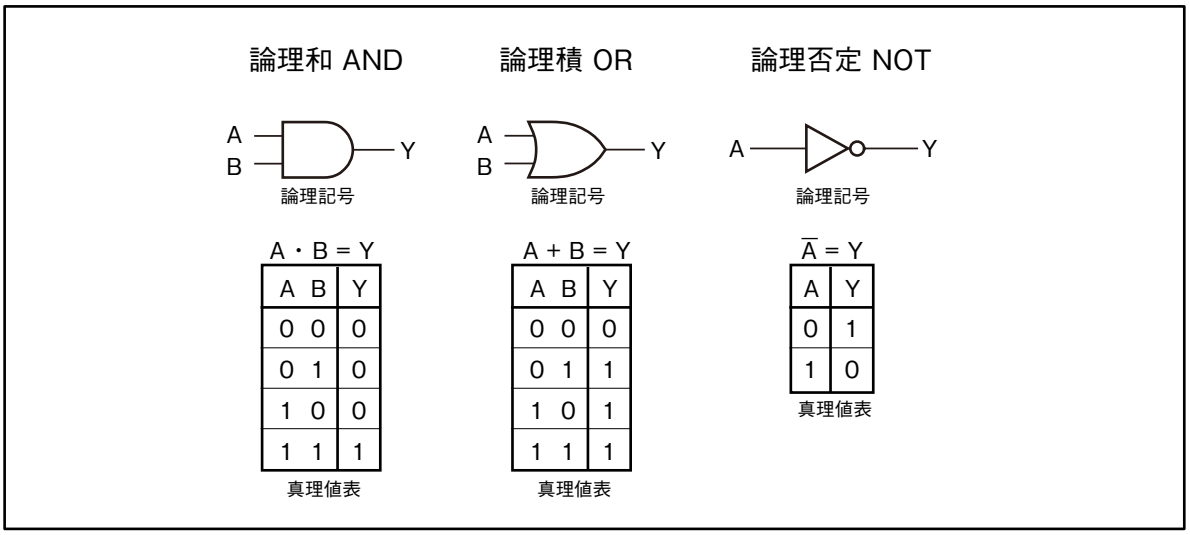


図2-1 基本ロジックのMIL記号と真理値表

論理回路では、その他のロジックのバリエーションも使います。これらのバリエーションはすべて AND、OR、NOT の 3 つのロジックを組み合わせてつくられたものです。(以下に示す回路以外にもバリエーションがあります。)

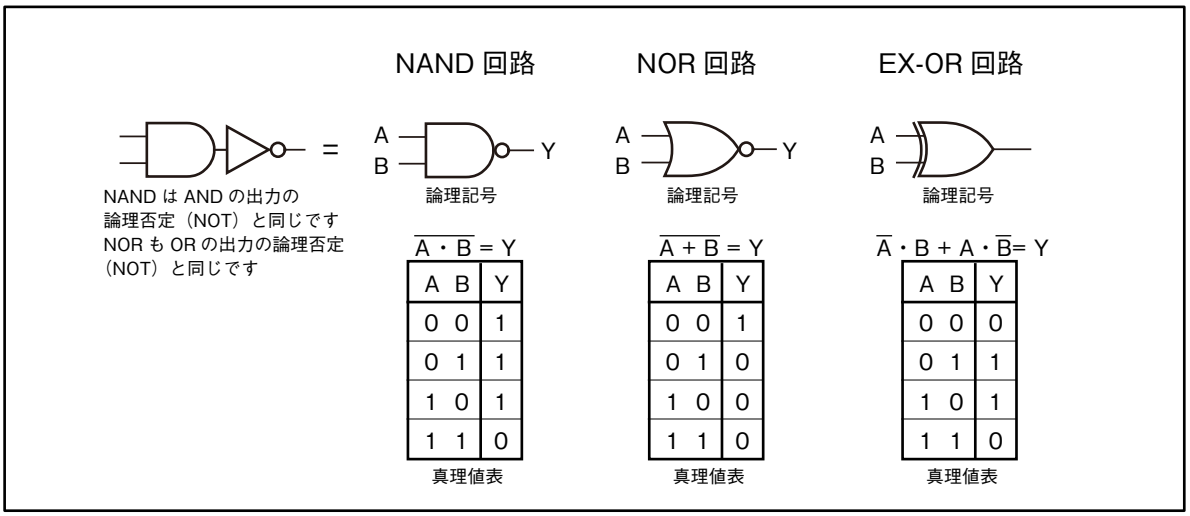


図2-2 ロジックのその他のバリエーション

Exercise 2.1①　基本ロジックの動作を確認しよう

論理和、論理積は、いわゆる四則演算の和、積とは違う論理です。
Applilet EZ PL で論理ゲートを配置し、LOGIC TRAINER+ に書込んで動作を確認してみましょう。
AND 回路の入力にスライドスイッチ SW1、SW2 を接続し、出力に LED1 を接続します。
OR 回路の入力に SW3、SW4 を、出力に LED2 を接続します。

真理値表 _2.1.1

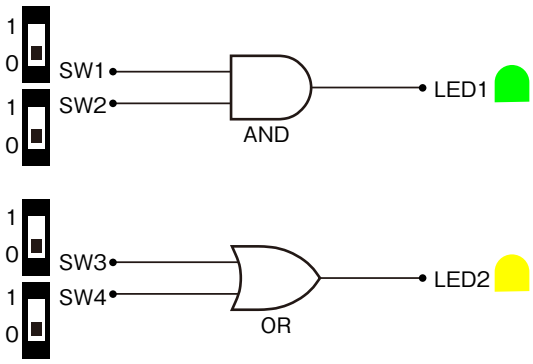
SW1	SW2	LED1
0	0	0（消灯）
0	1	0（消灯）
1	0	0（消灯）
1	1	1（点灯）

SW3	SW4	LED2
0	0	0（消灯）
0	1	1（点灯）
1	0	1（点灯）
1	1	1（点灯）

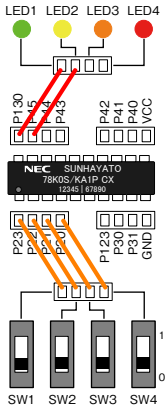
論理式 _2.1.1

LED1 = SW1 ・ SW2

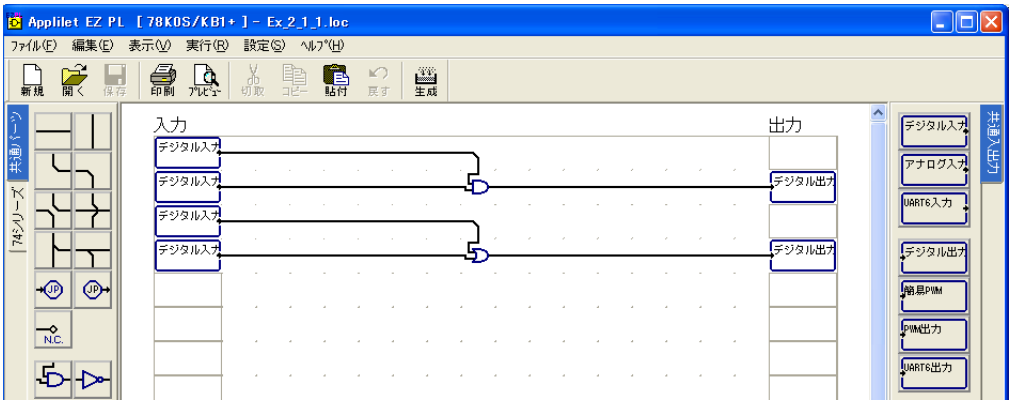
LED2 = SW3 ＋ SW4



接続図



LOGIC TRAINER+ の配線図



Applilet EZ PL のパネル配置

図2-3 Exercise2.1①

★ NOT、NAND、NOR、EX-OR 回路なども試してみましょう。

2.2 組み合わせ回路のつくりかた

ロジックを組み合わせて機能を持たせた回路を、組み合わせ回路といいます。組み合わせ回路をつくる手順を、以下の例題で示します。

> ＜例題＞
> LOGIC TRAINER+のスイッチ（SW1～4）のうち、いずれか3つのスイッチがONのときにLED1が点灯し、それ以外のスイッチの状態ではLEDは消灯するような組み合わせ回路をつくる

① 与えられる入力信号と得たい出力信号を、真理値表にする

SW 1～4 の考えられる組み合わせと、それに対する LED1 への信号を表にします。スイッチが ON なら 1 が入力され、LED1 を点灯するには 1 を出力することになります。

真理値表 _ 例題

SW1	SW2	SW3	SW4	LED1	
0	0	0	0	0（消灯）	
0	0	0	1	0（消灯）	
0	0	1	0	0（消灯）	
0	0	1	1	0（消灯）	
0	1	0	0	0（消灯）	
0	1	0	1	0（消灯）	
0	1	1	0	0（消灯））	
0	1	1	1	1（点灯）	SW1 ～ SW4 が 0111 のとき LED1 を点灯①
1	0	0	0	0（消灯）	
1	0	0	1	0（消灯）	
1	0	1	0	0（消灯）	
1	0	1	1	1（点灯）	SW1 ～ SW4 が 1011 のとき LED1 を点灯②
1	1	0	0	0（消灯）	
1	1	0	1	1（点灯）	SW1 ～ SW4 が 1101 のとき LED1 を点灯③
1	1	1	0	1（点灯）	SW1 ～ SW4 が 1110 のとき LED1 を点灯④
1	1	1	1	0（消灯）	

② 真理値表のアクティブとなる出力信号に注目し、アクティブ信号を出力するための論理を式にする

アクティブとは、ブール代数でいう「真」という意味です。この組み合わせ回路の目的は、「3 つのスイッチが ON なら LED を点灯する（＝ 1 を出力する）」なので、1 が「真」＝アクティブとなり、1 が出力されるような組み合わせのみ考えればよいことになります。

（回路の出力機器によっては､0 を出力するのが「真」＝アクティブとなる場合もあります。この場合はローアクティブと呼びます。）

LED1 に 1 を出力するようなスイッチの組み合わせを論理式にすると、以下のようになります。

真理値表の①より　LED1 = $\overline{SW1}$ ・ SW2 ・ SW3 ・ SW4（0 の場合は、記号の上に“ー”を付けます）

②より　LED1 = SW1 ・ $\overline{SW2}$ ・ SW3 ・ SW4

③より　LED1 = SW1 ・ SW2 ・ $\overline{SW3}$ ・ SW4

④より　LED1 = SW1 ・ SW2 ・ SW3 ・ $\overline{SW4}$

③ 論理式の簡単化を試みる

式は全て同じ LED1 に 1 を出力する論理で、いずれかの論理を満たせばよいので OR 論理でつなぐことができます。

式 1： LED1 ＝ $\overline{SW1}$ ・ SW2 ・ SW3 ・ SW4 ＋ SW1 ・ $\overline{SW2}$ ・ SW3 ・ SW4 ＋ SW1 ・ SW2 ・ $\overline{SW3}$ ・ SW4 ＋ SW1 ・ SW2 ・ SW3 ・ $\overline{SW4}$

式 1 の赤の下線部分について考えます。SW3･SW4 は共通項なので、まとめられます。

式 2：LED1 ＝ SW3･SW4・($\overline{SW1}$ **・ SW2 ＋ SW1 ・** $\overline{SW2}$ **)**

式 1 の青の下線部分についても、SW1･SW2 は共通項なので、まとめられます。

式 3：LED1 ＝ SW1･SW2・($\overline{SW3}$ **・ SW4 ＋ SW3 ・** $\overline{SW4}$ **)**

式 2、3 の緑の下線の式は、EX-OR 論理に置き換えられます。

式 4：LED1 ＝ SW3･SW4・(SW1 ⊕ SW2)

式 5：LED1 ＝ SW1･SW2・(SW3 ⊕ SW4)

さらに式 4、5 を 1 つにまとめると以下のようになります。

LED1 ＝ SW3･SW4・(SW1 ⊕ SW2) ＋ SW1･SW2・(SW3 ⊕ SW4)

④ 論理式を回路図にする

簡単化した論理式を回路図にすると、以下のようになります。

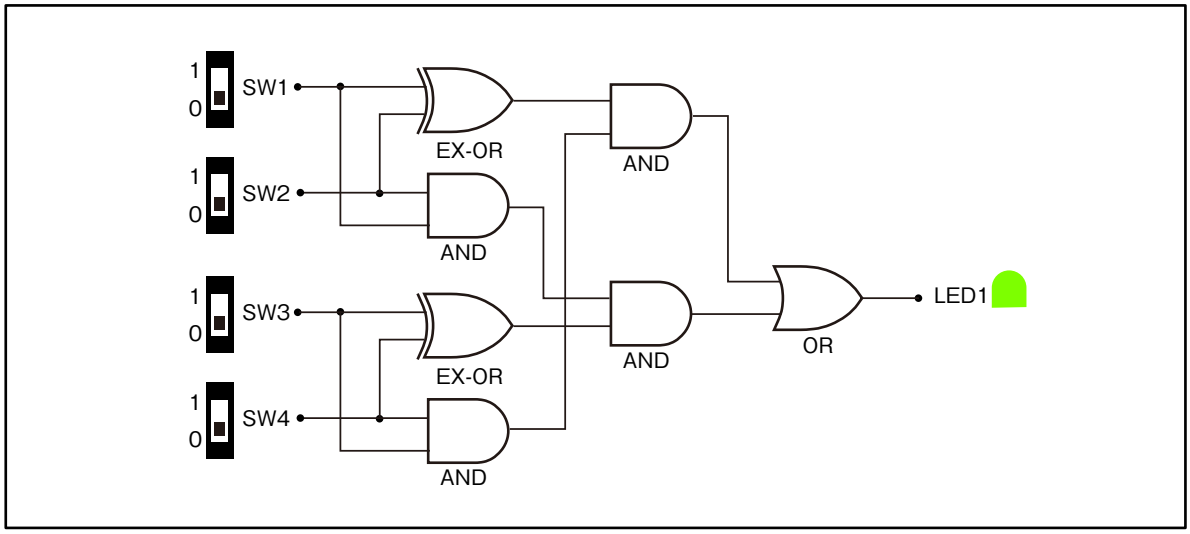


図2-4 論理式を回路図にする

論理式を簡単にする方法

この節では、ブール代数の定理にしたがって論理式を簡単にする方法を紹介していますが、カルノー図（Karnaugh map）や、クワイン・マクラスキー法（Quine-McCluskey algorithm）という方法もあります。ハードウェア記述言語で書いた場合はコンピュータが論理式を簡単化します。

2.3 セレクタ

セレクタとは、“選ぶもの”という意味ですが、論理回路の場合は複数の信号から一つを選択する回路を呼びます。複数の入力から一つを選ぶ回路はマルチプレクサ（multiplexer）とも呼びます。また、逆に複数の出力先から一つ選んで出力するのをデマルチプレクサ（demultiplexer）とも呼びます。

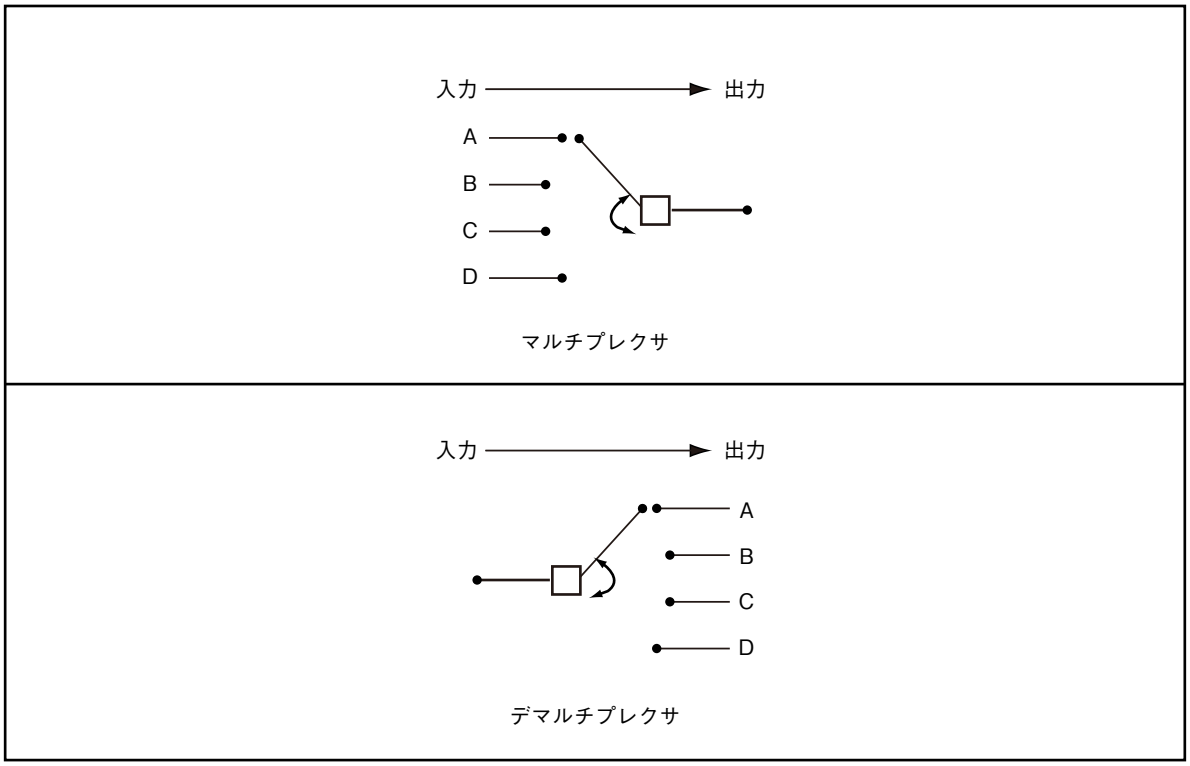


図2-5 マルチプレクサ、デマルチプレクサ

Exercise 2.3①　マルチプレクサをつくろう

SW2、SW3 の複数の入力信号を選択するセレクタ（マルチプレクサ）をつくってみましょう。どちらの信号を選択するかは、SW1 で操作します。SW1 が 1 のとき、SW2 の値を LED1 に出力し、SW1 が 0 のとき、SW3 の値を LED1 に出力します。

真理値表 _2.3.1

SW1	SW2	SW3	LED1	
0	*	0	0（消灯）	
0	*	1	1（点灯）	← SW1 が 0 で、なおかつ SW3 が 1 のとき点灯①
1	0	*	0（消灯）	
1	1	*	1（点灯）	← SW1 が 1 で、なおかつ SW2 が 1 のとき点灯②

＊＝don't care（どの値でも出力に影響を与えないという意味です）

真理値表の、LED1 が点灯する条件について考えます。

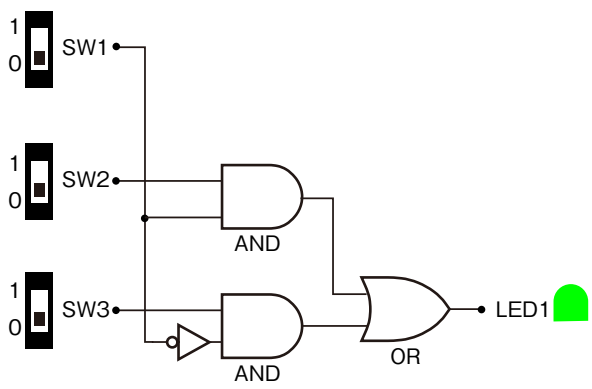
真理値表の①より　LED1 ＝ $\overline{\text{SW1}}$・SW3

②より　LED1 ＝ SW1・SW2

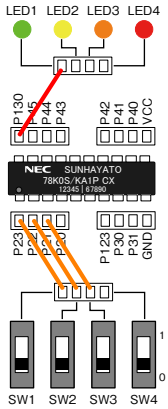
LED1 を点灯するには、いずれかの論理が成り立てばいいので、2 つの式は OR 論理でつなげられます。

論理式 _2.3.1

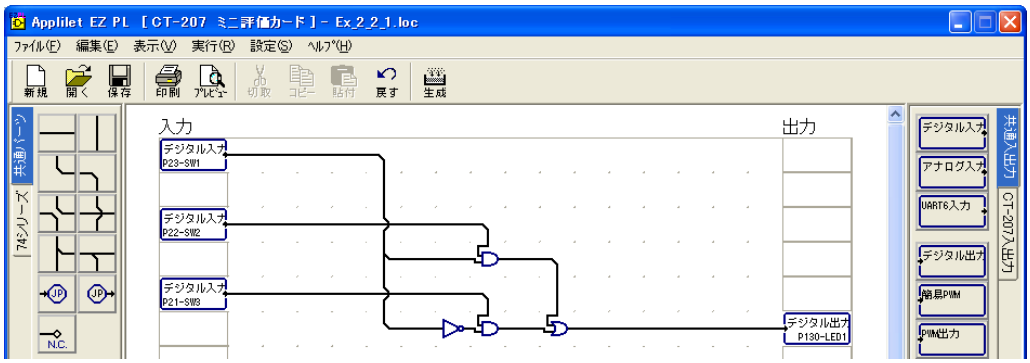
LED1 ＝（$\overline{\text{SW1}}$・SW3）＋（SW1・SW2）



接続図



LOGIC TRAINER+ の配線図



Applilet EZ PL のパネル配置

図2-6 Exercise 2.2 ①

Exercise 2.3②　デマルチプレクサをつくろう

SW3 の一つの入力信号を、LED1、LED2、LED3、LED4 の複数の出力に割り当てるセレクタ（デマルチプレクサ）をつくってみましょう。どの出力先（LED）に割り当てるかは SW1、SW2 の組み合わせで決めることとします。

真理値表 _2.3.2

SW1	SW2	SW3	LED1	LED2	LED3	LED4	
1	1	0	0（消灯）	0（消灯）	0（消灯）	0（消灯）	← SW1、SW2 が 11 のとき、LED1 を点灯　①
1	1	1	1（点灯）	0（消灯）	0（消灯）	0（消灯）	
0	1	0	0（消灯）	0（消灯）	0（消灯）	0（消灯）	← SW1、SW2 が 01 のとき、LED2 を点灯　②
0	1	1	0（消灯）	1（点灯）	0（消灯）	0（消灯）	
1	0	0	0（消灯）	0（消灯）	0（消灯）	0（消灯）	← SW1、SW2 が 10 のとき、LED3 を点灯　③
1	0	1	0（消灯）	0（消灯）	1（点灯）	0（消灯）	
0	0	0	0（消灯）	0（消灯）	0（消灯）	0（消灯）	← SW1、SW2 が 00 のとき、LED4 を点灯　④
0	0	1	0（消灯）	0（消灯）	0（消灯）	1（点灯）	

真理値表の、SW3 の状態が LED1 ～ 4 に出力される SW1、SW2 の組み合わせを考えます。

LED1 に出力される組み合わせ → SW1・SW2
LED2 に出力される組み合わせ → $\overline{SW1}$・SW2
LED3 に出力される組み合わせ → SW1・$\overline{SW2}$
LED4 に出力される組み合わせ → $\overline{SW1}$・$\overline{SW2}$

さらに SW1、SW2 の条件がそろったときだけ SW3 の状態がそのまま各 LED に出力され、それ以外の組み合わせのときは消灯（0 を出力）なので、SW1、SW2 の論理式を、SW3 と AND をとります。

論理式 _2.3.2

LED1 ＝（SW1・SW2）・SW3
LED2 ＝（$\overline{SW1}$・SW2）・SW3
LED3 ＝（SW1・$\overline{SW2}$）・SW3
LED4 ＝（$\overline{SW1}$・$\overline{SW2}$）・SW3

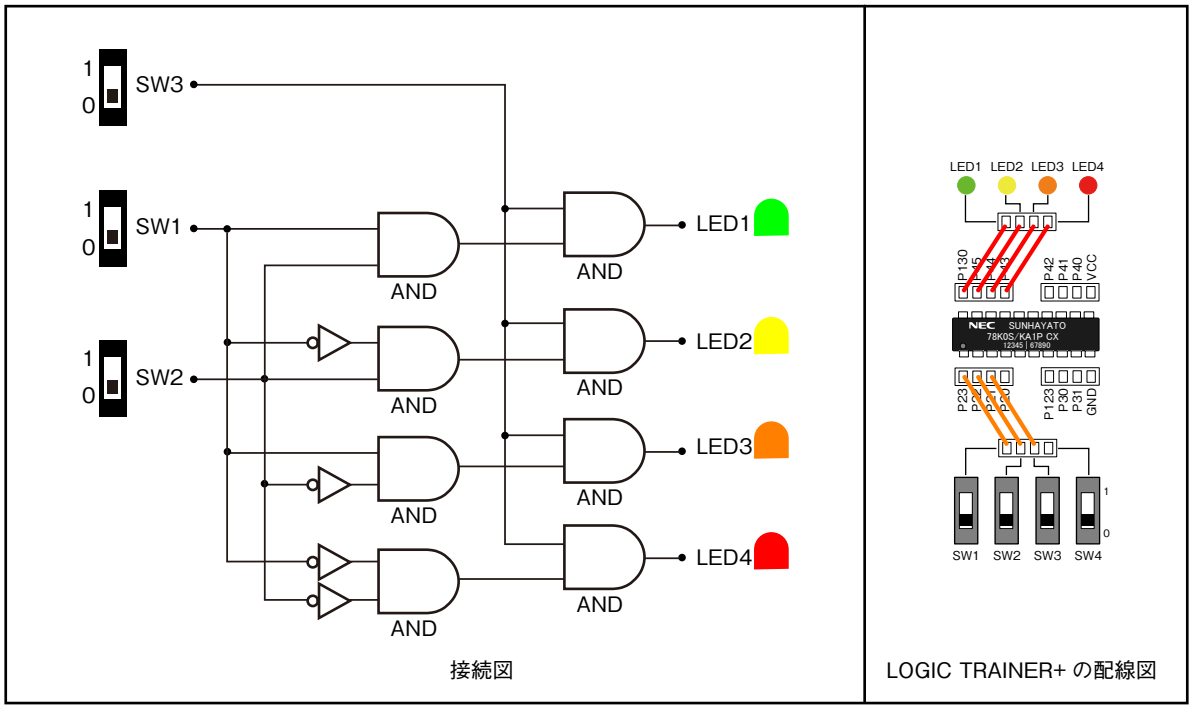
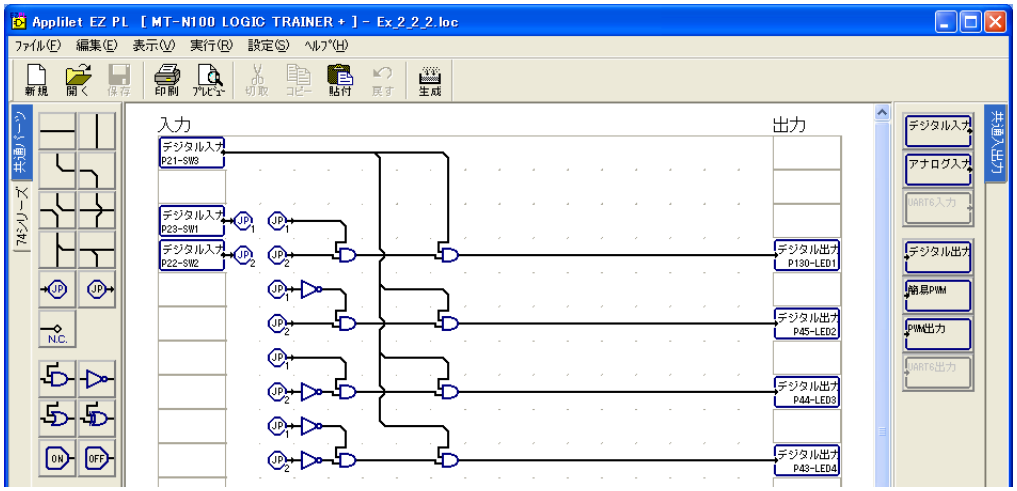
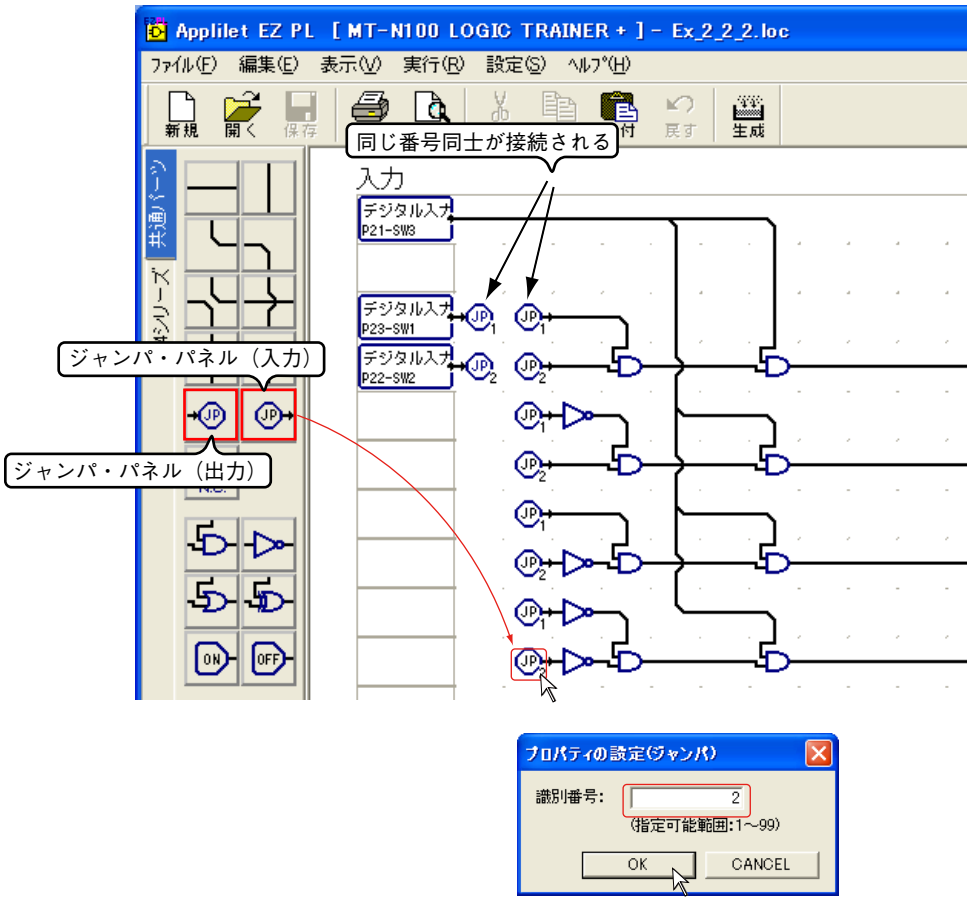


図2-7 Exercise 2.3 ②

この例題では配線が少し複雑なので、Applilet EZ PL では「ジャンパ」パネルを使ってみましょう。「ジャンパ」パネルを使うと、同じ識別番号の「ジャンパ」パネルが接続されている離れた線同士が接続していることになります。識別番号はプロパティで設定します。「ジャンパ」パネルは便利な機能ですが、使いすぎると信号の流れがわかりにくくなりますので注意しましょう。



Applilet EZ PL のパネル配置



「ジャンパ」パネルの配置と設定

図2-8 Applilet EZ PL 「ジャンパ」パネル

2.4 エンコーダ、デコーダ

エンコーダ（符号器）とは、“入力値をある一定の規則で符号化（ある意味を持つ特殊な値に加工）するもの”という意味です。デコーダ（復号器）は逆に符号化したデータを元に戻すものを指します。

コンピュータでは、メモリに書かれた命令を読み出し内容に従って動作しますが、命令を解釈して制御信号を出力する回路もデコーダになります。

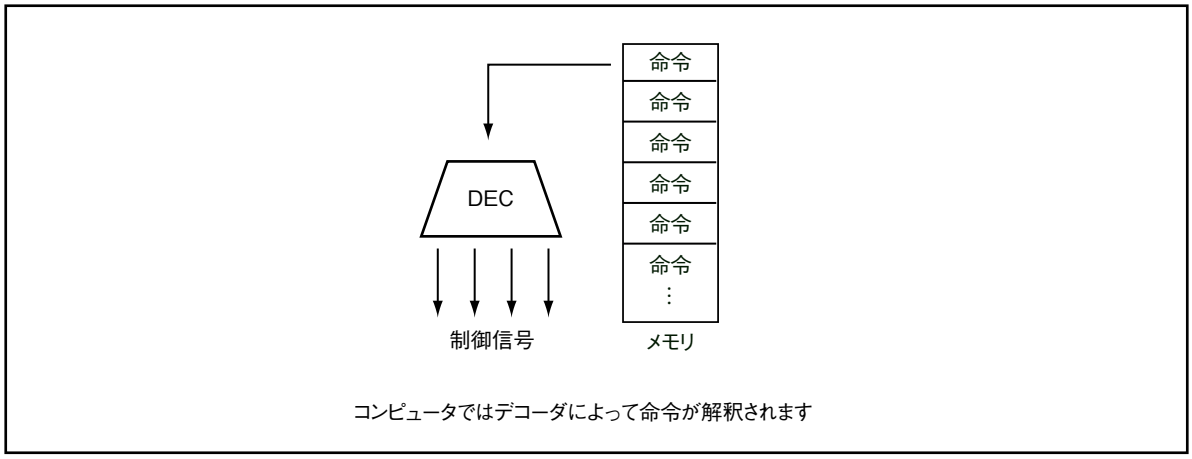


図2-9 コンピュータのデコーダ

2進数について（1）

論理回路では、「1」「0」の２つの値を扱いますが、これを２進数の数字として計算をしたり情報をあらわしたりします。２進数と 16 進数は、桁上がりのきりが同じで都合が良いので、２進数をわかりやすくかつ短く表すのに 16 進数で表記されることがあります。

また、論理回路では１本の信号線で「1」「0」を表しますが、この単位をビットと呼びます。

８ビットコンピュータというと、８ビットのデータを一度に扱えるコンピュータということになりますが、８ビットで表すことのできる２進数の数値は 0000 0000 ～ 1111 1111 ということになります。これを 16 進数で表すと、00 ～ FF となります。10 進数では 0 ～ 255 となります。

10、16、２進数対応表

10 進数	16 進数	２進数	
0	0	0	
1	1	1	
2	2	10	
3	3	11	
4	4	100	
5	5	101	
6	6	110	
7	7	111	
8	8	1000	
9	9	1001	
10	A	1010	
11	B	1011	
12	C	1100	
13	D	1101	
14	E	1110	
15	F	1111	
16	10	10000	← 16 進数と２進数の桁上がりが同じ

Exercise 2.4 ①　エンコーダをつくろう

SW1 ～ 4 の状態をエンコードし、LED2 ～ 4 に 3 桁の 2 進数で表示するエンコーダをつくってみましょう。
LED4 を 3 桁の最下位桁とします。
SW4 は 1、SW3 は 2、SW2 は 3、SW1 は 4 を表すこととします。
まず、SW1 ～ 4 のうちの一つだけが 1 のときのみを考え、それ以外の組み合わせは考えないこととします。

真理値表 _2.4.1

SW1 (4)	SW2 (3)	SW3 (2)	SW4 (1)	LED2	LED3	LED4	
0	0	0	0	0	0	0	
0	0	0	1	0	0	1	← SW4 が 1 のとき、LED4 が点灯　①
0	0	1	0	0	1	0	← SW3 が 1 のとき、LED3 が点灯　②
0	1	0	0	0	1	1	← SW2 が 1 のとき、LED3、LED4 が点灯　③
1	0	0	0	1	0	0	← SW1 が 1 のとき、LED2 が点灯　④

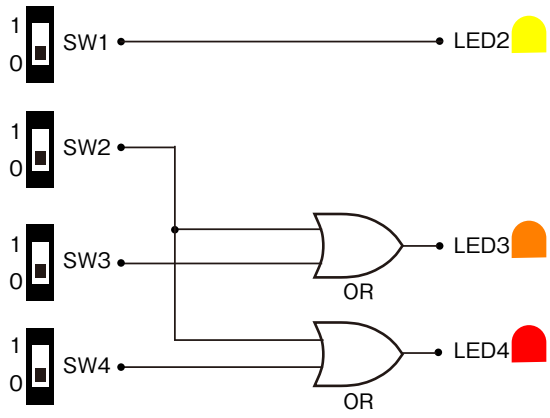
真理値表の、各 LED が点灯する条件について考えます。

論理式 _2.4.1

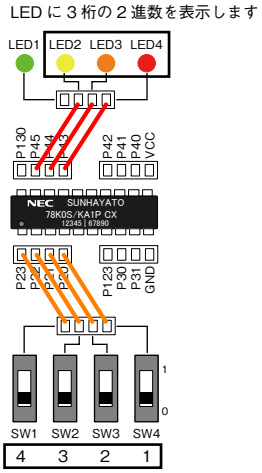
LED4 = SW4 + SW2　(①、③)

LED3 = SW3 + SW2　(②、③)

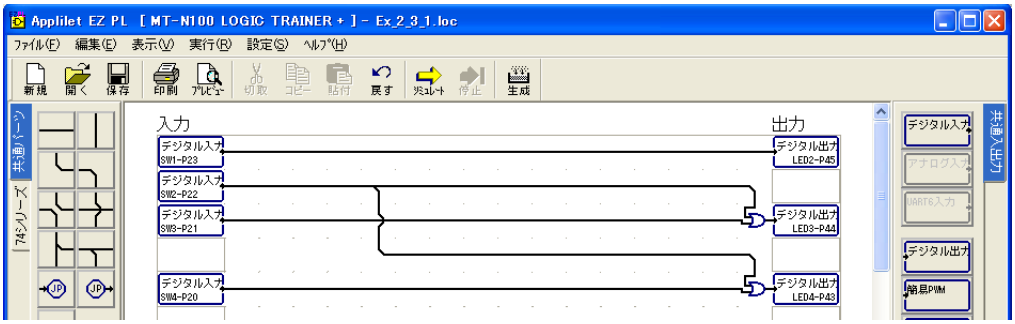
LED2 = SW1　(④)



接続図



SW に数字がふられています
LOGIC TRAINER+ の配線図



Applilet EZ PL のパネル配置

図2-10 Exercise 2.3 ①

Exercise 2.4 ②　エンコーダの入力に優先順位をつけよう

Exercise 2.4 ①では、SW1 ～ 4 のうちの一つだけが 1 のときのみを考え、それ以外の組み合わせの場合は考えませんでした。次はそれ以外の組み合わせを考え、スイッチに優先順位を付けてみましょう。

優先度は SW1 が一番が高く、2、3、4 と順に下がることとします。

真理値表 _2.4.2

SW1 (4)	SW2 (3)	SW3 (2)	SW4 (1)	LED2	LED3	LED4		
0	0	0	0	0	0	0		
0	0	0	1	0	0	1	← SW1 ～ SW4 が 0001 のとき、LED4 が点灯	①
0	0	1	*	0	1	0	← SW1 ～ SW3 が 001 のとき、LED3 が点灯	②
0	1	*		0	1	1	← SW1、SW2 が 01 のとき、LED3、LED4 が点灯	③
1	*			1	0	0	← SW1 が 1 のとき、LED2 が点灯	④

＊＝ don't care（どの値でも出力に影響を与えないという意味です

真理値表の各 LED が点灯する条件について考えます。() 内は真理値表の各条件を表します。

＜ LED4 が点灯する条件＞

LED4 = $\overline{SW1}$・$\overline{SW2}$・$\overline{SW3}$・SW4　（真理値表①より）
LED4 = $\overline{SW1}$・SW2　（真理値表③より　SW3、SW4 は影響を与えないので式に入れる必要はありません）

2 つの式をまとめると以下の式になります。

LED4 = ($\overline{SW1}$・$\overline{SW2}$・$\overline{SW3}$・SW4) + ($\overline{SW1}$・SW2)

さらにブール代数の分配の法則より以下の式になります。

LED4 = $\overline{SW1}$・($\overline{SW2}$・$\overline{SW3}$・SW4 + SW2)

さらにブール代数の吸収の法則より以下の式になります。

LED4 = $\overline{SW1}$・($\overline{SW3}$・SW4 + SW2)

該当するブール代数の定理

＜分配の法則＞
A・(B + C) = A・B + A・C

＜吸収の法則＞
A + $\overline{A}$・B = A + B

＜ LED3 が点灯する条件＞

LED3 = $\overline{SW1}$・$\overline{SW2}$・SW3（真理値表② より SW4 は影響を与えないので式に入れる必要はありません）
LED3 = $\overline{SW1}$・SW2　（真理値表③より SW3、SW4 は影響を与えないので式に入れる必要はありません）

2 つの式をまとめると以下の式になります。

LED3 = $\overline{SW1}$・($\overline{SW2}$・SW3) + ($\overline{SW1}$ + SW2)

さらにブール代数の分配の法則より以下の式になります。

LED3 = $\overline{SW1}$・($\overline{SW2}$・SW3 + SW2)

さらにブール代数の吸収の法則より以下の式になります。

LED3 = $\overline{SW1}$・(SW3 + SW2)

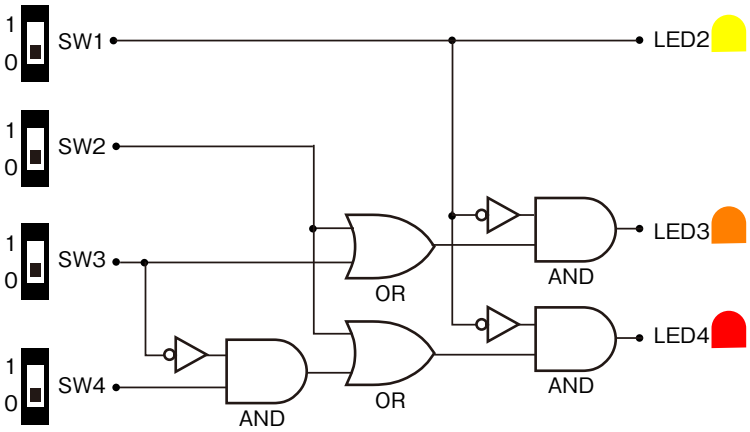
＜ LED2 が点灯する条件＞

LED2 = SW1（④ SW2 ～ SW4 は影響を与えないので式に入れる必要はありません）

以上の計算で、3 つの LED を点灯させる式は 3 つにまとまります。

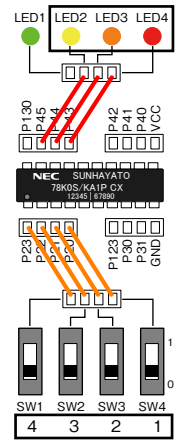
論理式 _2.4.2

LED4 = $\overline{SW1}$・($\overline{SW3}$・SW4 + SW2)
LED3 = $\overline{SW1}$・(SW3 + SW2)
LED2 = SW1



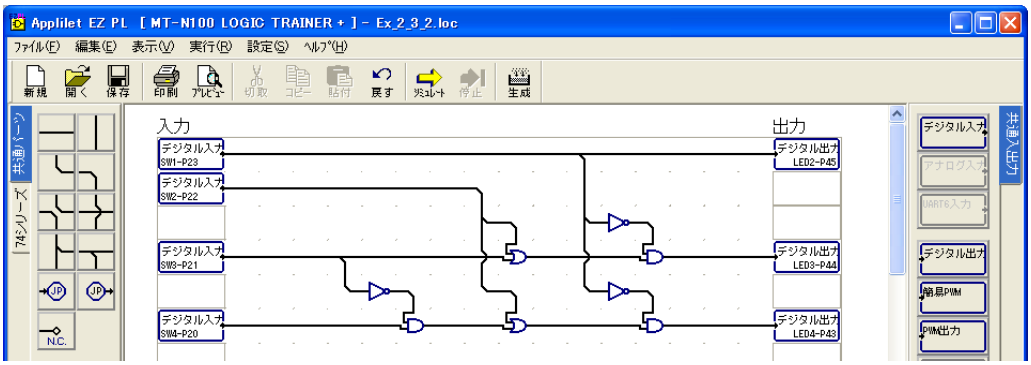
接続図

LED に 3 桁の 2 進数を表示します



SW に数字がふられています

LOGIC TRAINER+ の配線図

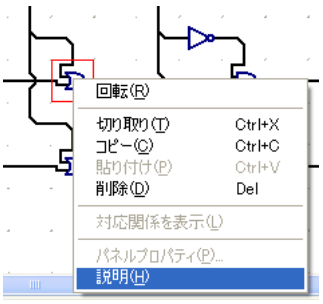


Applilet EZ PL のパネル配置

図2-11 Exercise 2.4 ②

Applilet EZ PL：パネルについて調べる

Applilet EZ PL の処理パネルを選択した状態で右クリックし「説明」をクリックすると、
メイン・ウィンドウ下に詳細な説明が表示されます。
パネルが論理計算を行う場合は真理値表も表示されます。



Exercise 2.4 ③　デコーダをつくろう

SW2 ～ 4 から入力される値を 3 桁の 2 進数のデータとみなし、数字の 1 ～ 4 に割り当てられた LED を点灯するデコーダをつくってみましょう。SW4 を 3 桁の最下位桁とします。
デコードする数字は LED1 ～ 4 で表すことのできる 0 ～ 4 まで（2 進数で 000 ～ 100 まで）とします。それ以外の SW2 ～ 4 の組み合わせが入力された場合は、すべての LED が消灯することとします。

真理値表 _2.4.3

SW2	SW3	SW4	LED1	LED2	LED3	LED4	
0	0	0	0	0	0	0	
0	0	1	0	0	0	1	← SW2 ～ SW4 が 001 のとき、LED4 が点灯　①
0	1	0	0	0	1	0	← SW2 ～ SW4 が 010 のとき、LED3 が点灯　②
0	1	1	0	1	0	0	← SW2 ～ SW4 が 011 のとき、LED2 が点灯　③
1	0	0	1	0	0	0	← SW2 ～ SW4 が 100 のとき、LED1 が点灯　④
1	0	1	0	0	0	0	
1	1	0	0	0	0	0	
1	1	1	0	0	0	0	

真理値表の各 LED が点灯する条件について考えます。（）内は真理値表の各条件を表します。

論理式 _2.4.3_01

LED4 = $\overline{SW2}$ ・ $\overline{SW3}$ ・ SW4　(①)
LED3 = $\overline{SW2}$ ・ SW3 ・ $\overline{SW4}$　(②)
LED2 = $\overline{SW2}$ ・ SW3 ・ SW4　(③)
LED1 = SW2 ・ $\overline{SW3}$ ・ $\overline{SW4}$　(④)

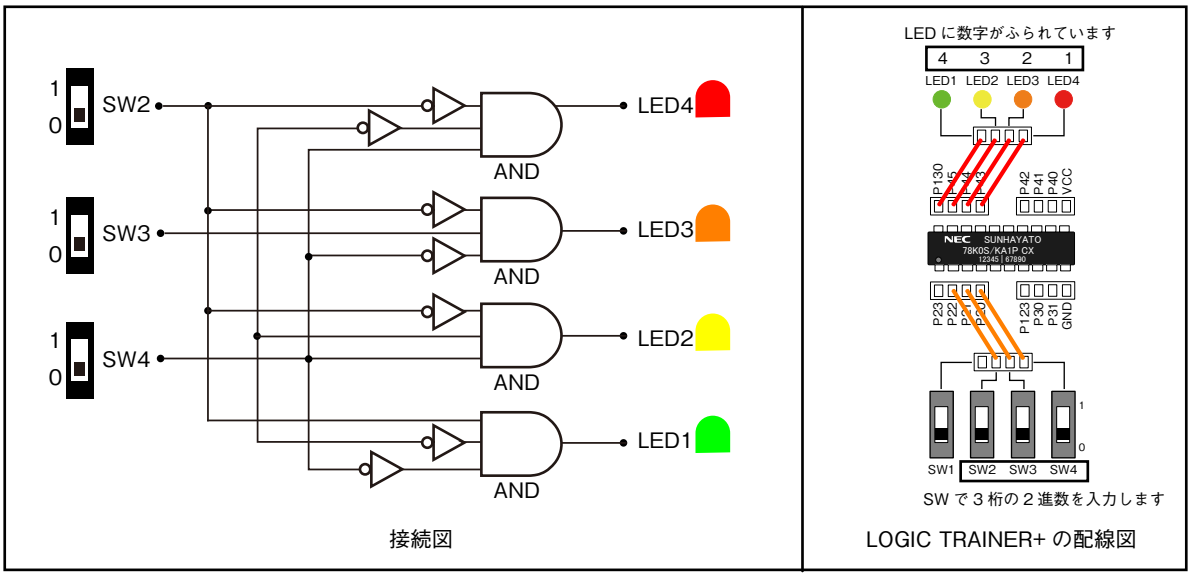
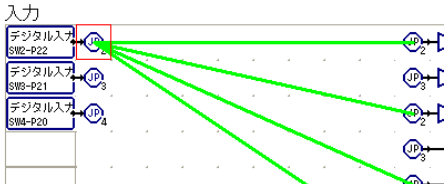
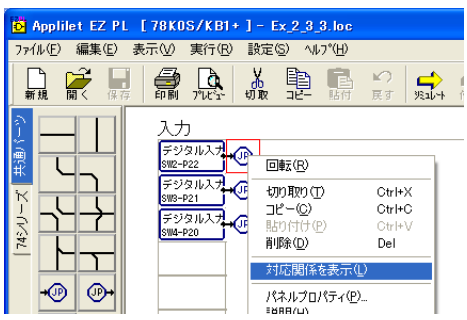


図2-12 Exercise 2.4 ③ 接続図

Applilet EZ PL：ジャンパパネルの関係を調べる

ジャンパパネルを右クリックし、表示されるメニューの「対応関係を表示」をクリックすると、選択したジャンパが実際にどこにつながっているかが表示され、確認することができます。





Applilet EZ PL で回路図をつくる場合は、3 入力の AND パネルがないので、以下の式に置きかえます。

論理式 _2.4.3_02

LED4 = (SW2 ・ SW3) ・ SW4

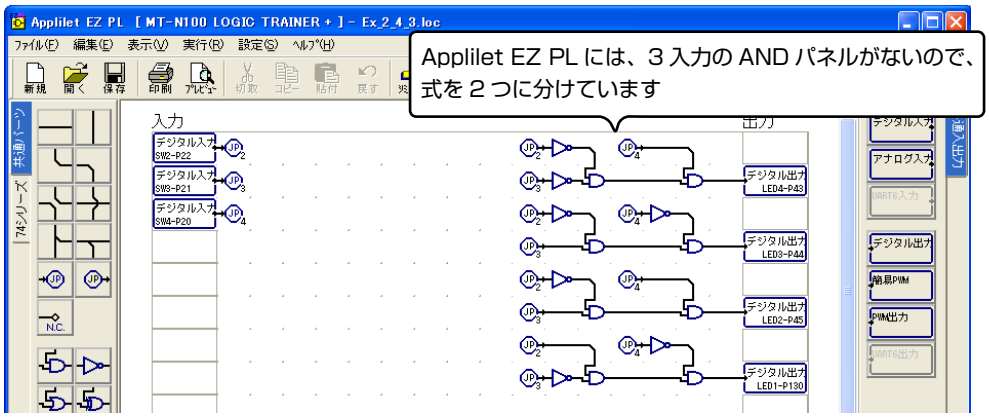
LED3 = (SW2 ・ SW3) ・ SW4

LED2 = (SW2 ・ SW3) ・ SW4

LED1 = (SW2 ・ SW3) ・ SW4

該当するブール代数の定理

＜結合の法則＞
A ・ (B ・ C) = (A ・ B) ・ C

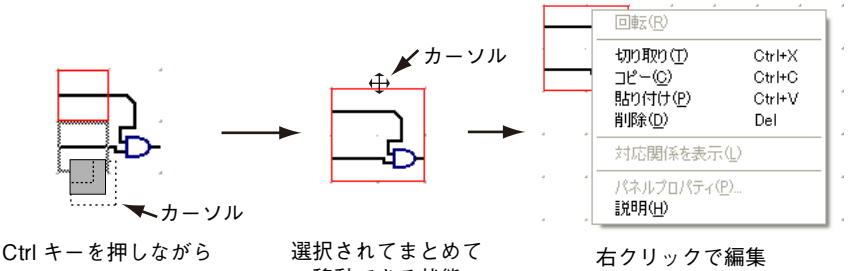


Applilet EZ PL のパネル配置

図2-13 Exercise 2.4 ③ パネル配置

Applilet EZ PL：複数のパネルをまとめて移動、コピーする

入出力パネルや処理パネルを複数コピーあるいは移動する場合は、ひとつのパネルを選択した（左クリックを押したまま）状態で「Ctrl」キーを押しながら、さらにドラッグすると複数のパネルが赤い枠で囲まれ、選択できます。赤い枠で囲まれた状態でいったんクリックを解消し、ふたたび赤い枠の上にカーソルをのせるとカーソルが十字の形になり、その状態でドラッグすると赤い枠ごと移動できます。また、赤い枠に囲まれた状態で右クリックすると各編集メニューが出ます。



Ctrl キーを押しながら
ドラッグする

選択されてまとめて
移動できる状態

右クリックで編集

2.5 加算器

加算器は、2進数の足し算をする回路をいいます。

コンピュータには、必ず計算をおこなう論理演算装置（ALU：Arithmetic Logic Unit）という回路があり、加算器も ALU の機能の一部です。

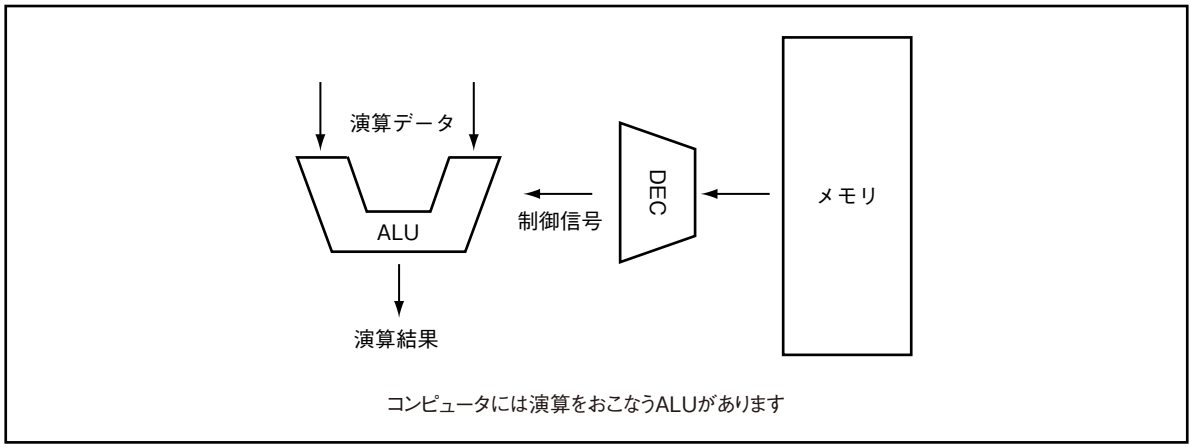


図2-14 コンピュータのALU

加算器には、入力に桁上がり（Carry:キャリーといいます）がなく、下位の桁上がりを足さない半加算器（HA:Half Adder）と、入力に桁上がりがあり、下位の桁上がりも足す全加算器（FA：Full Adder）があります。たとえば2桁の足し算の場合、下位桁は半加算器で足し算をし、上位桁は全加算器で、下位桁の桁上がりも入れた足し算をします。

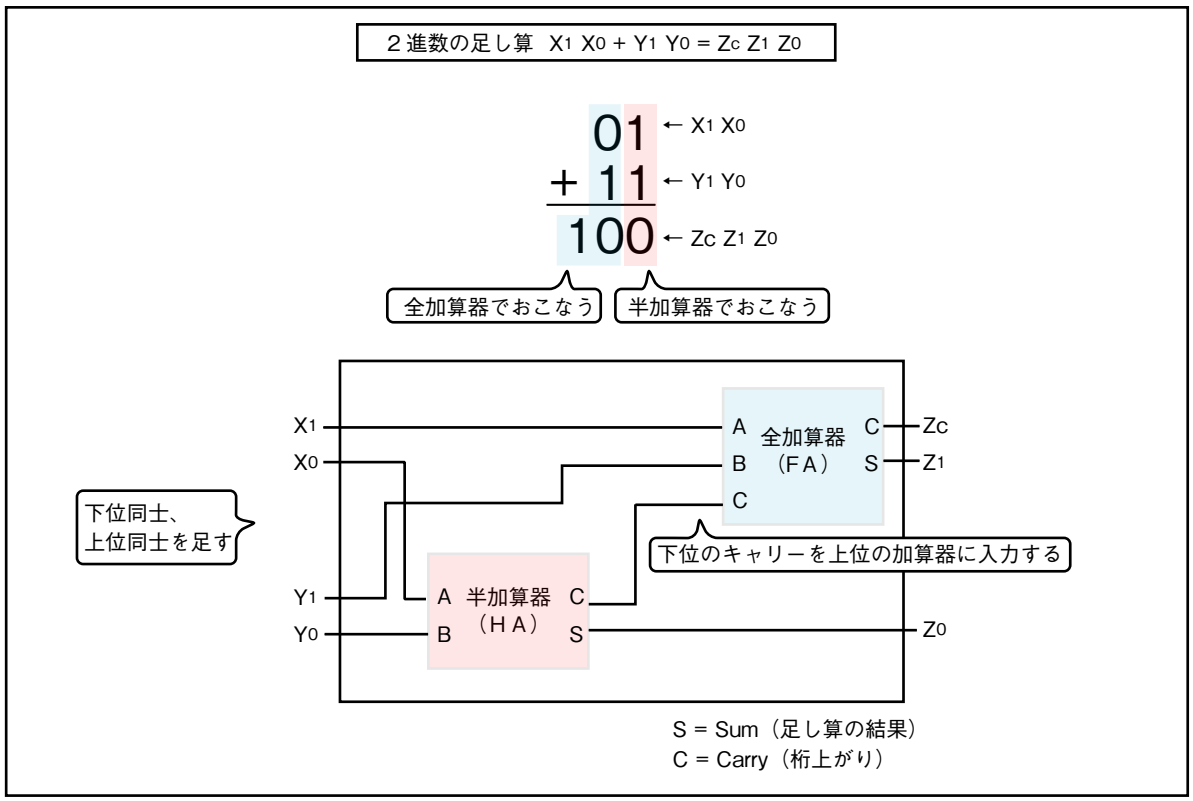


図2-15 半加算器と全加算器で2桁の計算をする

半加算器は、真理値表より以下の回路になります。

半加算器の真理値表

X1	X0	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

半加算器の論理式

$S = \overline{X1} \cdot X0 + X1 \cdot \overline{X0}$ ←この式はEX-ORゲートと同じなので置き換えられます

$C = X1 \cdot X0$

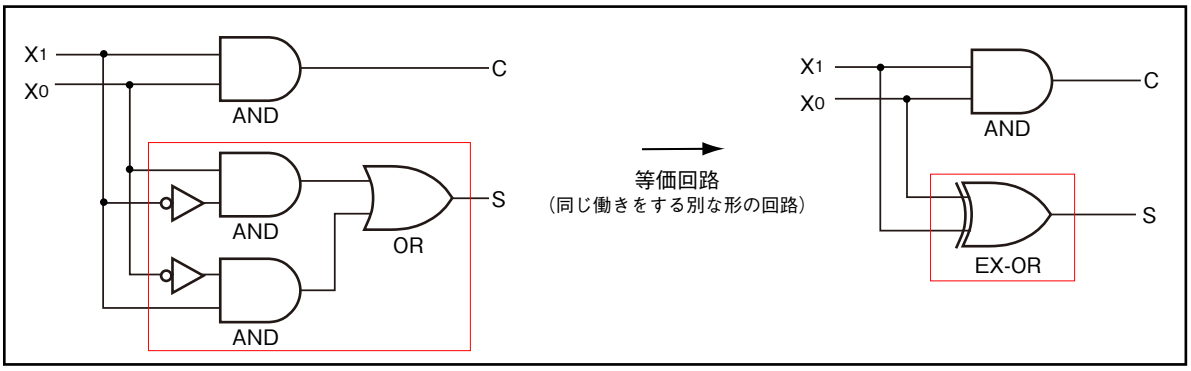


図2-16 半加算器の論理回路

また、全加算器は、半加算器を２つ組み合わせてつくることができます。

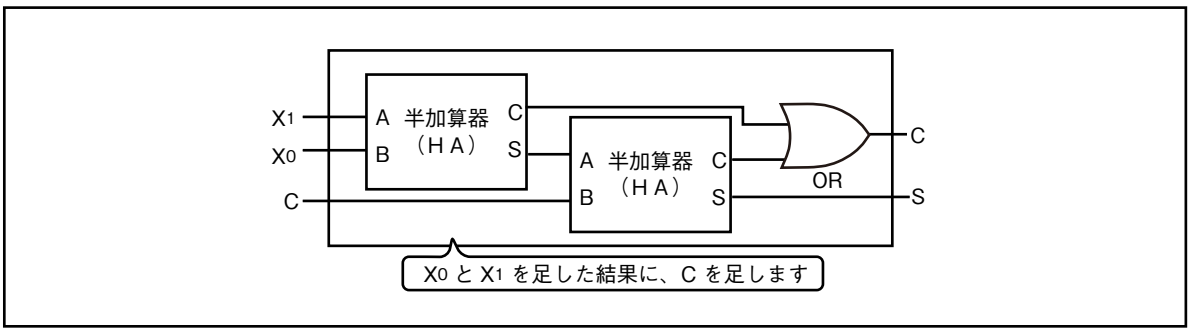


図2-17 半加算器を2つ使って全加算器をつくる

2進数について（2）

２進数は、０→１の次は桁上がりになります。以下に２進数の足し算を示します。

２進数の足し算（１桁）

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 ← 上位の 1 が桁上がり
```

２進数の足し算（２桁）

```
00 + 00 = 00    01 + 01 = 10    10 + 10 = 100
00 + 01 = 01    01 + 10 = 11    10 + 11 = 101
00 + 10 = 10    01 + 11 = 100   11 + 11 = 110
00 + 11 = 11          ↑               ↑
                  上位の 1 が桁上がり
```

Exercise 2.5 ①　加算器をつくろう

SW1、2 と SW3、4 を、それぞれ 2 桁の 2 進数データとみなし、足し算をして LED4 ～ LED2 に 2 進数で表示する加算回路をつくりましょう。SW2、SW4 を 2 桁の下位桁とします。

真理値表 _2.5.1

SW1	SW2	SW3	SW4	LED2	LED3	LED4	
0	0	0	0	0	0	0	00 + 00 = 000
0	0	0	1	0	0	1	00 + 01 = 001
0	0	1	0	0	1	0	00 + 10 = 010
0	0	1	1	0	1	1	00 + 11 = 011
0	1	0	1	0	1	0	01 + 01 = 010
0	1	1	0	0	1	1	01 + 10 = 011
0	1	1	1	1	0	0	01 + 11 = 100
1	0	1	0	1	1	0	10 + 10 = 110
1	0	1	1	1	0	1	10 + 11 = 101
1	1	1	1	1	1	0	11 + 11 = 110

この演習では論理式から回路を導かずに、前ページの説明を元に半加算器を組み合わせてつくります。

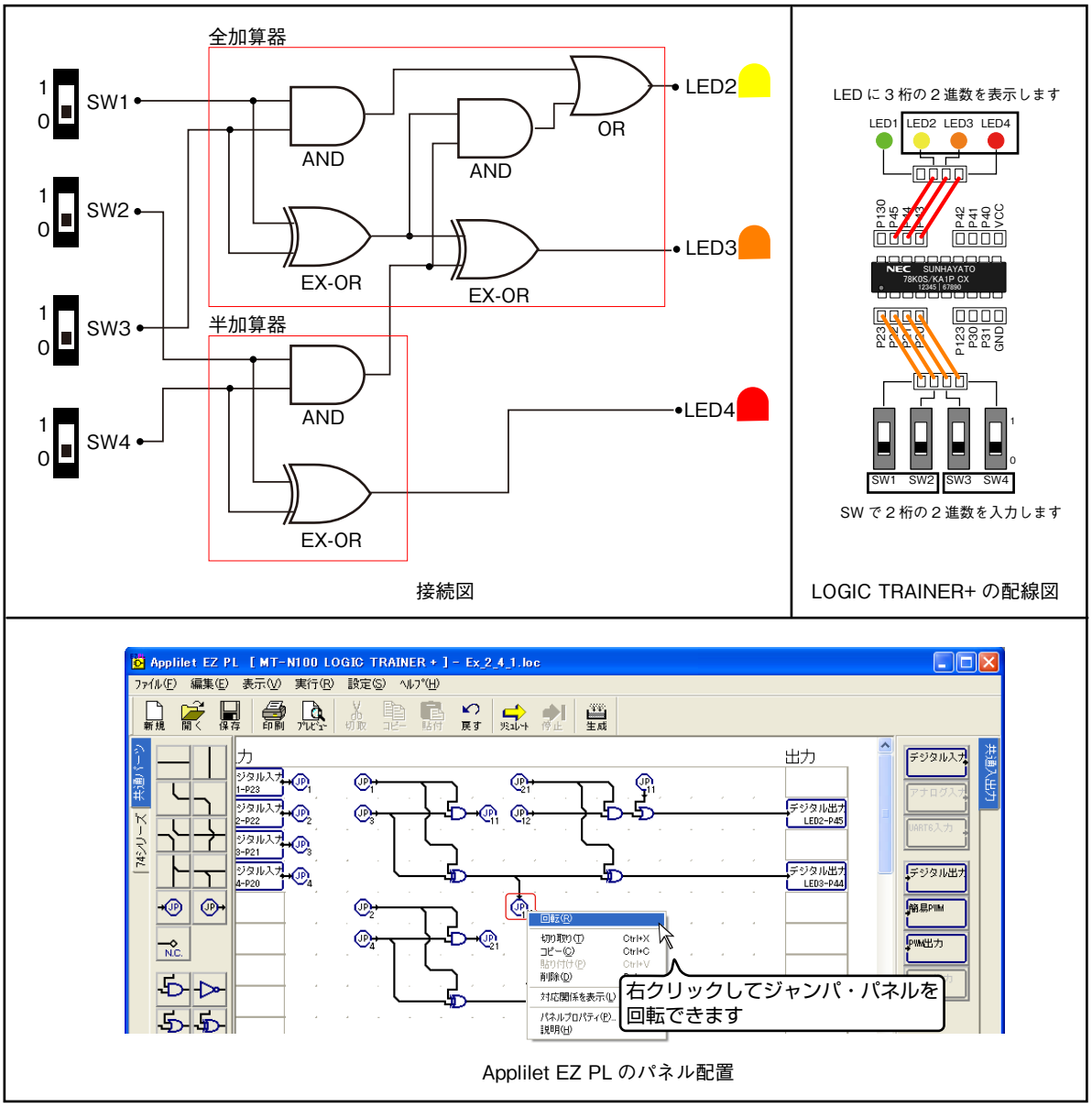


図2-18 Exercise 2.5 ①

3. 組み合わせ回路と順序回路でつくる

この章では、組み合わせ回路と順序回路を組み合わせた論理回路を紹介します。順序回路を使うとビットの状態を記憶することができます。ゲート回路と順序回路の組み合わせは、論理回路の基本ともいえるでしょう。

3.1 基本の記憶回路（フリップフロップ）
3.2 シフトレジスタ
3.3 非同期式カウンタ
3.4 同期式カウンタ

3.1 基本の順序回路（フリップフロップ）

データを保存する機能を持つ回路を順序回路といいます。保存されている（以前の）状態が決まっていて、現在の入力信号の状態と合わせてから出力が決まることから順序回路と呼ばれます。順序回路の基本となる 1 ビットのデータを保存する回路をフリップフロップ（図中では FF と表記します）といいます。

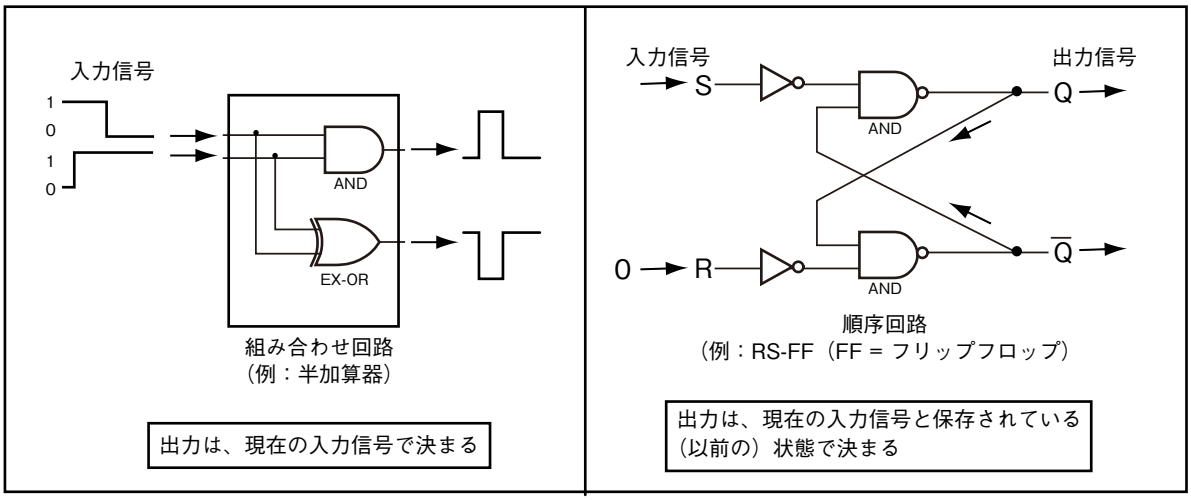


図3-1 組み合わせ回路と順序回路

組み合わせ回路の場合は、現在の入力信号の状態のみについて考えましたが、順序回路の場合は、入力信号の現在の状態と、それまでに保存されていた状態という時間的な要素が加わることになります。

順序回路の入力信号が保存されるタイミングは、入力信号で決まります。フリップフロップの種類によっては、タイミング専用の信号を入力するものや、入力信号の状態が変化するタイミングで保存するものなどがあります。

3.1.1 フリップフロップの動作原理

RS（リセット・セット）フリップフロップを例に、動作についてみていきましょう。

RS フリップフロップは、S（セット）、R（リセット）信号を入力に持つフリップフロップです。出力は Q と Q の反転信号である $\overline{Q}$ です。

① 現在 0 を保存していて（Q = 0）、S より 1 を、R より 0 を入力した場合

回路は、Q = 1、$\overline{Q}$ = 0 の状態で安定します。

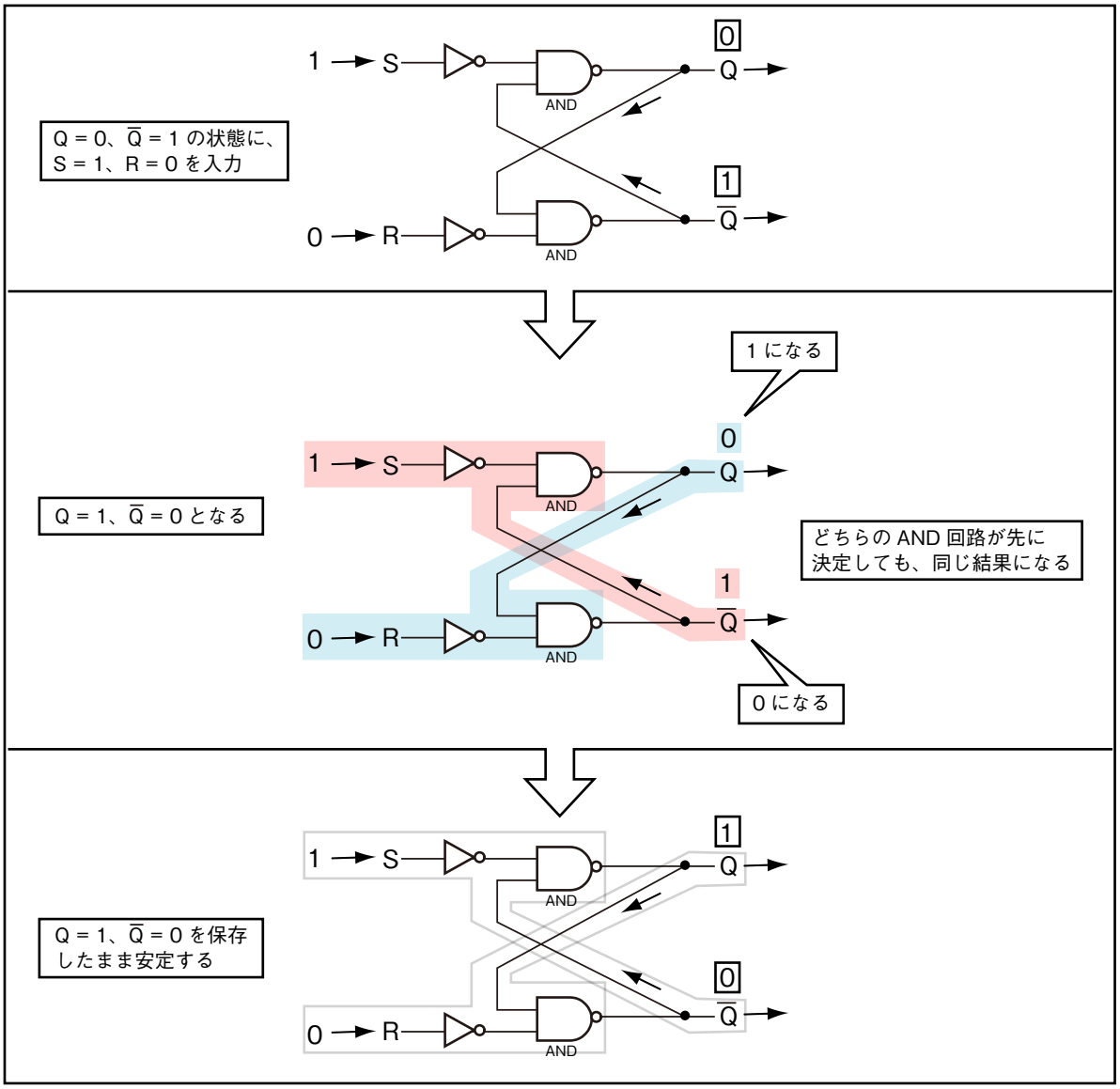


図3-2 RS-FFの動作①

② **現在 1 を保存していて（Q = 1、Q = 0）、S より 0 を、R より 1 を入力した場合**

回路は Q = 0、$\overline{Q}$ = 1 のまま安定します。

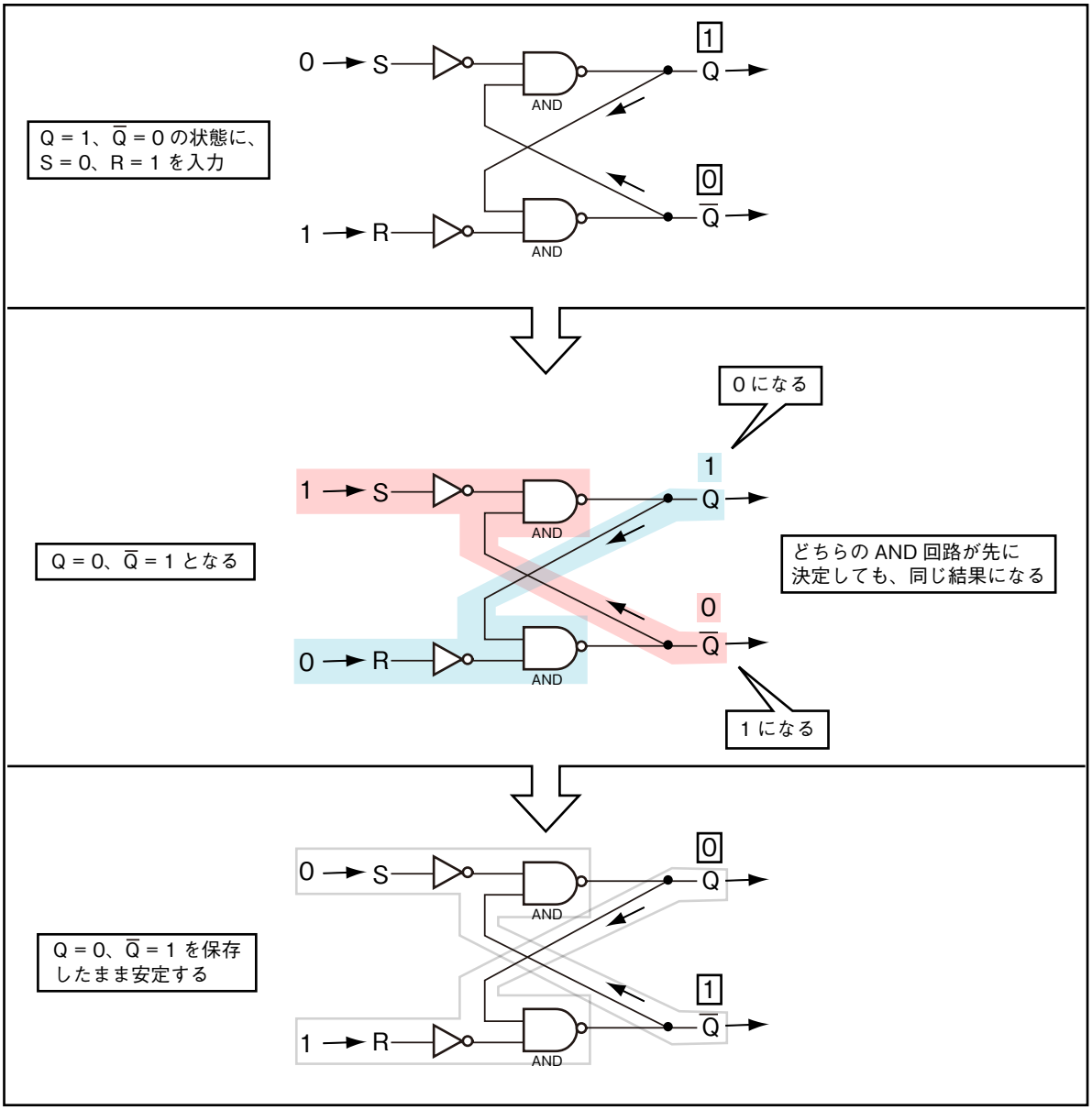


図3-3 RS－FFの動作②

③ 現在 0 を保存していて（Q = 1）、S より 1 を、R より 1 を入力した場合

回路は Q = 1、$\overline{Q}$ = 1 のまま安定しますが、この状態で、S、R とも 0 を入力した場合、どちらの AND 回路の結果が先に決まるかで、結果が異なります。
この場合、S と R に 1 を入力すると、次の動作の結果が予測できないということになります。

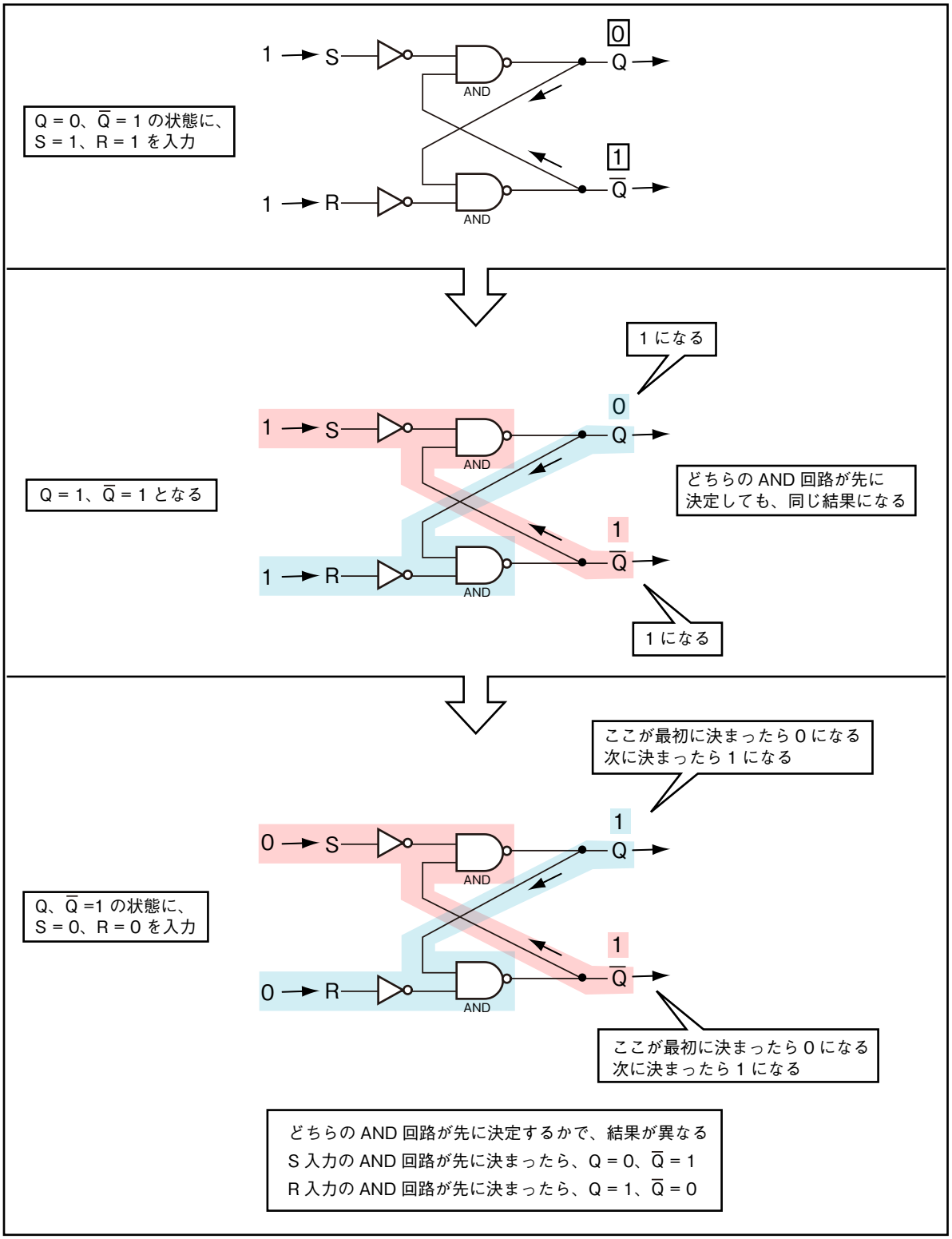


図3-4 RS－FFの動作③

以上の①～③までの動作をまとめると、以下の真理値表になります。

RS-FFの真理値表

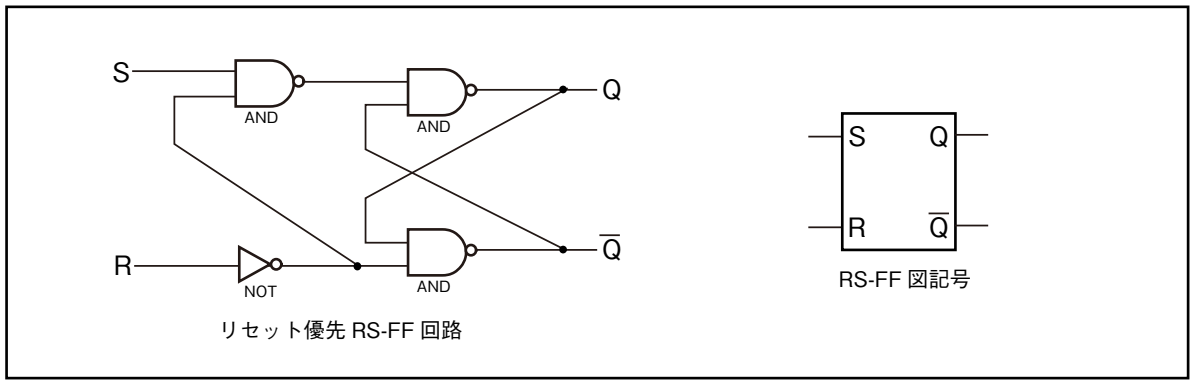
S	R	Q	$\overline{Q}$	動作
0	0	Q	$\overline{Q}$	保持
0	1	0	1	リセット
1	0	1	0	セット
1	1	不定		禁止

Applilet EZ PL の RS フリップフロップは、禁止入力（R = 1、S = 1）があっても安定に動作するリセット優先 RS フリップフロップです。

リセット優先 RS-FFの真理値表

S	R	Q	$\overline{Q}$	動作
0	0	Q	$\overline{Q}$	保持
*	1	0	1	リセット
1	0	1	0	セット
*	1	0	1	リセット

* = don't care（どの値でも出力に影響を与えないという意味です）



リセット優先 RS-FF 回路

図3-5 リセット優先RS-FFの回路

3.3.2 フリップフロップの種類

ディジタル回路では、その他のフリップフロップのバリエーションも使います。これらのフリップフロップはすべて RS フリップフロップを元につくられたものです。

（以下に示すフリップフロップ以外にもバリエーションがあります。）

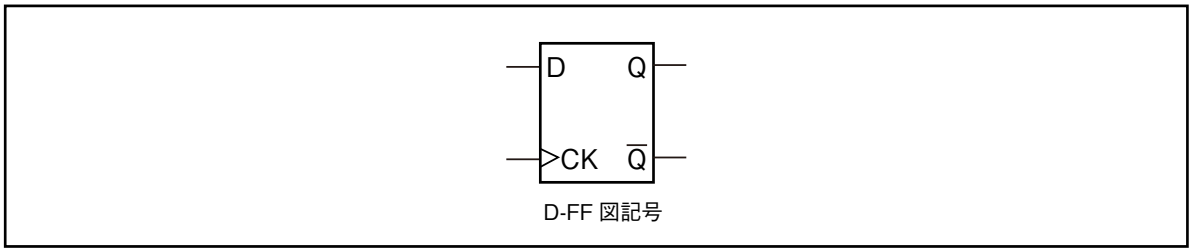
■ D（Delay）フリップフロップ

CK 信号の立ち上がりのタイミングで D 信号の状態を取り入れます。CK 信号の立ち上がり以外では、内容を保持したままです。

D-FF の真理値表

D	CK	Q	$\overline{Q}$
0	↑	0	1
1	↑	1	0
*	↑以外	保持	

*** ＝ don't care（どの値でも出力に影響を与えないという意味です）**
↑＝信号の 0 から 1 への変化（立ち上がり）を表します



D-FF 図記号

図3-8 Dフリップフロップ

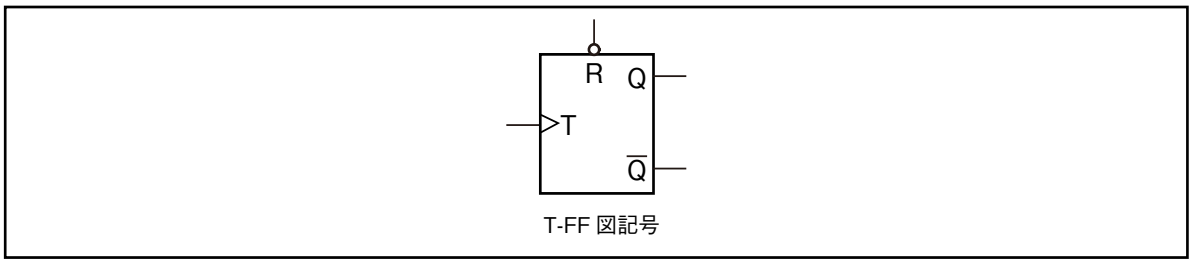
■ T（Trigger）フリップフロップ

T 信号の立ち上がり毎に、内容が反転します。R 信号が 0 になると、リセットされます。

T-FF の真理値表

T	R	Q	$\overline{Q}$
↑	1	反転	
↑以外	1	保持	
*	0	0	0

*** ＝ don't care（どの値でも出力に影響を与えないという意味です）**
↑＝信号の 0 から 1 への変化（立ち上がり）を表します



T-FF 図記号

図3-9 Tフリップフロップ

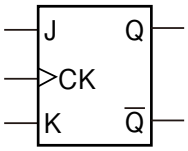
■ JKフリップフロップ

CK 信号の立ち上がりのタイミングで、J、K 信号の組み合わせで変化します。J、K 信号が 0、1 のときはリセットを、1、0 のときはセットをします。0、0 のときは保持、1、1 のときは反転します。

JK-FF の真理値表

J	K	CK	Q	$\overline{Q}$
0	0	↑	保持	
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	反転	
*	*	↑以外	保持	

＊＝ don't care（どの値でも出力に影響を与えないという意味です）
↑＝入力信号の 0 から 1 への変化（立ち上がり）を表します



JK-FF 図記号

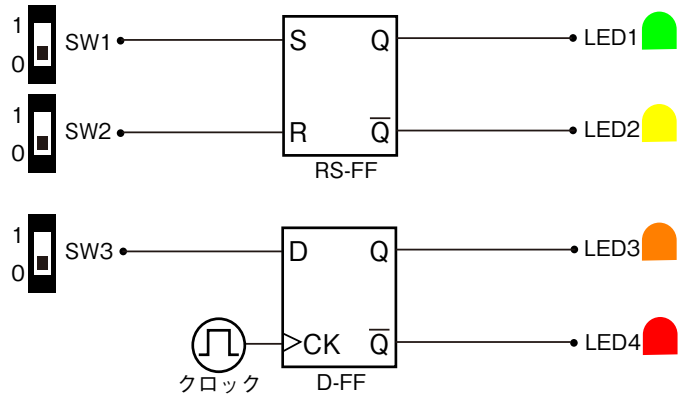
図3-10：JKフリップフロップ

フリップフロップは通常、論理式では表さないで真理値表、タイムチャートなどで表します。

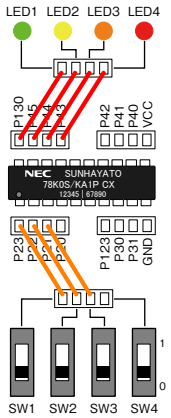
Exercise 3.1 ①　フリップフロップの動作を確認しよう

Applilet EZ PL の各「フリップフロップ」パネルの動作を確認してみましょう。パネルのプロパティで種類と初期値を設定します。CK 入力には「クロック」パネルを使います。

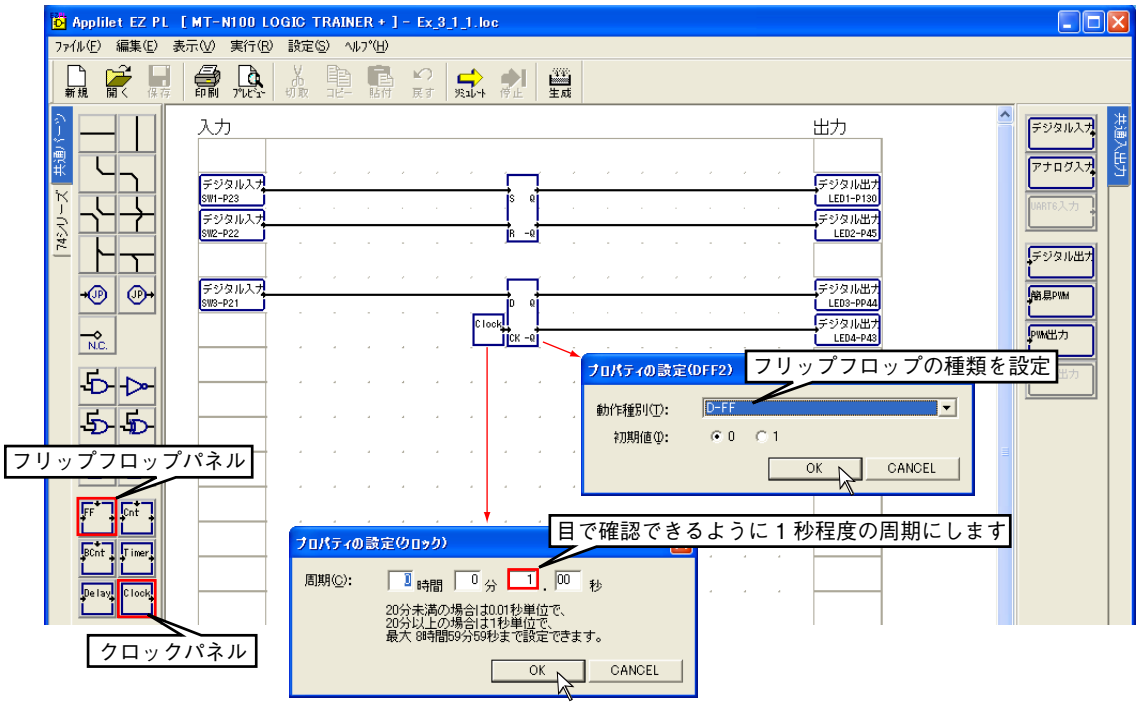
CK 以外の入力にはスライドスイッチ（SW1、2）を、出力 Q、$\overline{Q}$ には LED を接続します。



接続図

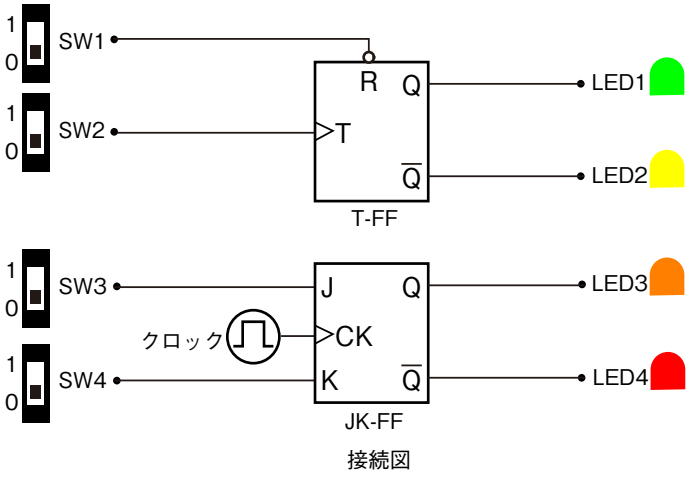


LOGIC TRAINER+ の配線図

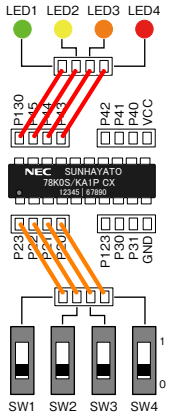


Applilet EZ PL のパネル配置

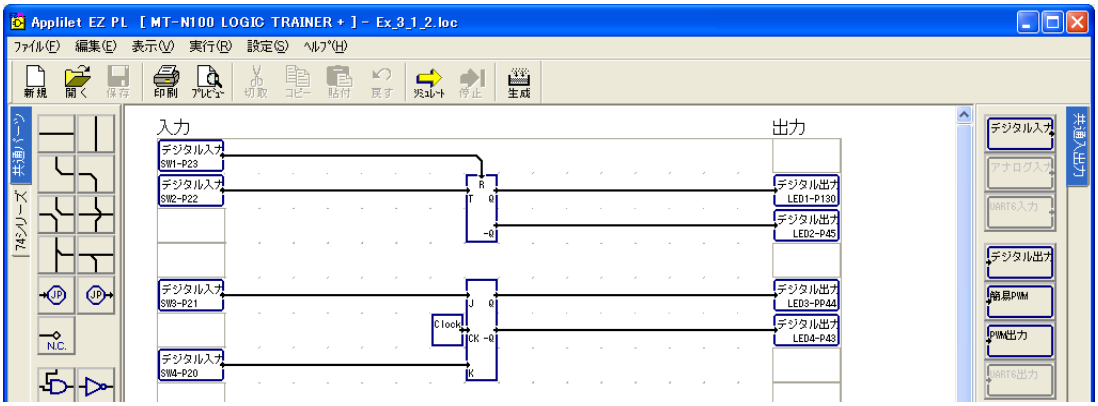
図3-11 RSフリップフロップとDフリップフロップ



接続図



LOGIC TRAINER+ の配線図



Applilet EZ PL のパネル配置

図3-12 TフリップフロップとJKフリップフロップ

3.2 レジスタ

レジスタはデータを一時的に記憶しておく記憶装置です。コンピュータではデータの演算や加工をする際に、かならずレジスタを使います。レジスタの数はコンピュータの規模によりますが、ほとんどのコンピュータは演算専用として最もよく使われるアキュムレータ（Accumlator）レジスタを中心に、複数のレジスタを持っています。

「32 ビットコンピュータ」などとビットの数でコンピュータの性能を表しますが、一度に扱うことのできるデータのビット数を表しており、大抵の場合、コンピュータ内部のレジスタの幅と一致します。

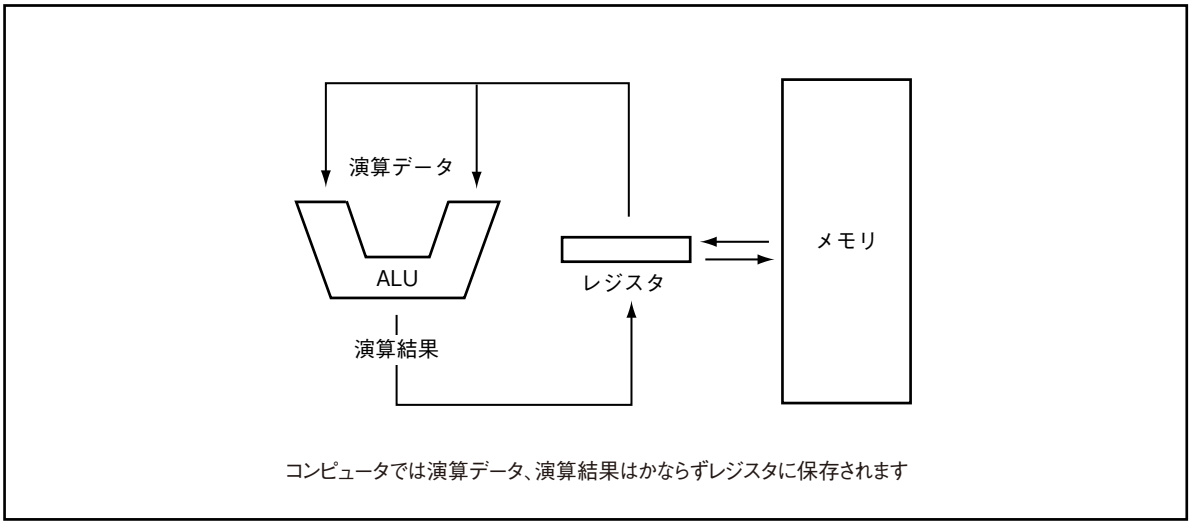


図3-13 コンピュータのレジスタ

LOGIC TRAINER+ に搭載している 8 ビットマイクロコンピュータ 78K0S/KA1+ には 8 ビットのレジスタが 8 個あります。2 つのレジスタを組み合わせて 16 ビットのレジスタとして使うこともできるようになっています。

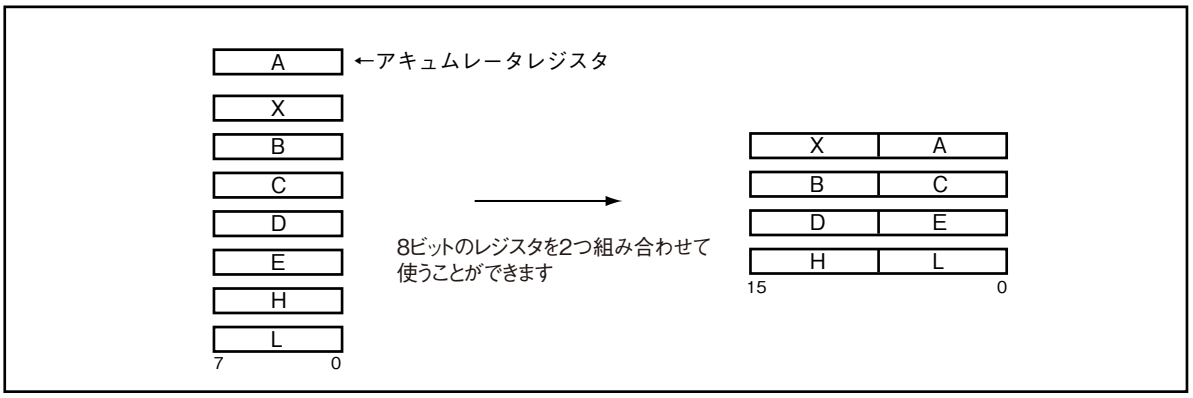


図3-14 78K0S/KA1+のレジスタ構成

Exercise 3.2 ①　シフトレジスタをつくろう

Ｄフリップフロップを使って、4ビットのシフトレジスタをつくってみましょう。シフトレジスタは、保持しているデータを1ビットづつシフトする機能のあるレジスタです。レジスタの各ビットの出力にLEDを接続して、データのシフトを確認できるようにします。

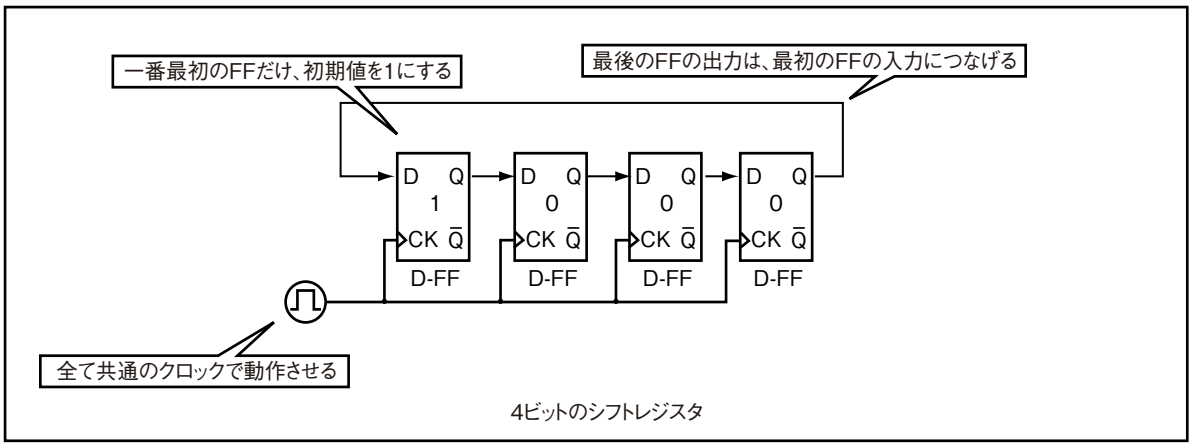


図3-15 4ビットのシフトレジスタの仕様

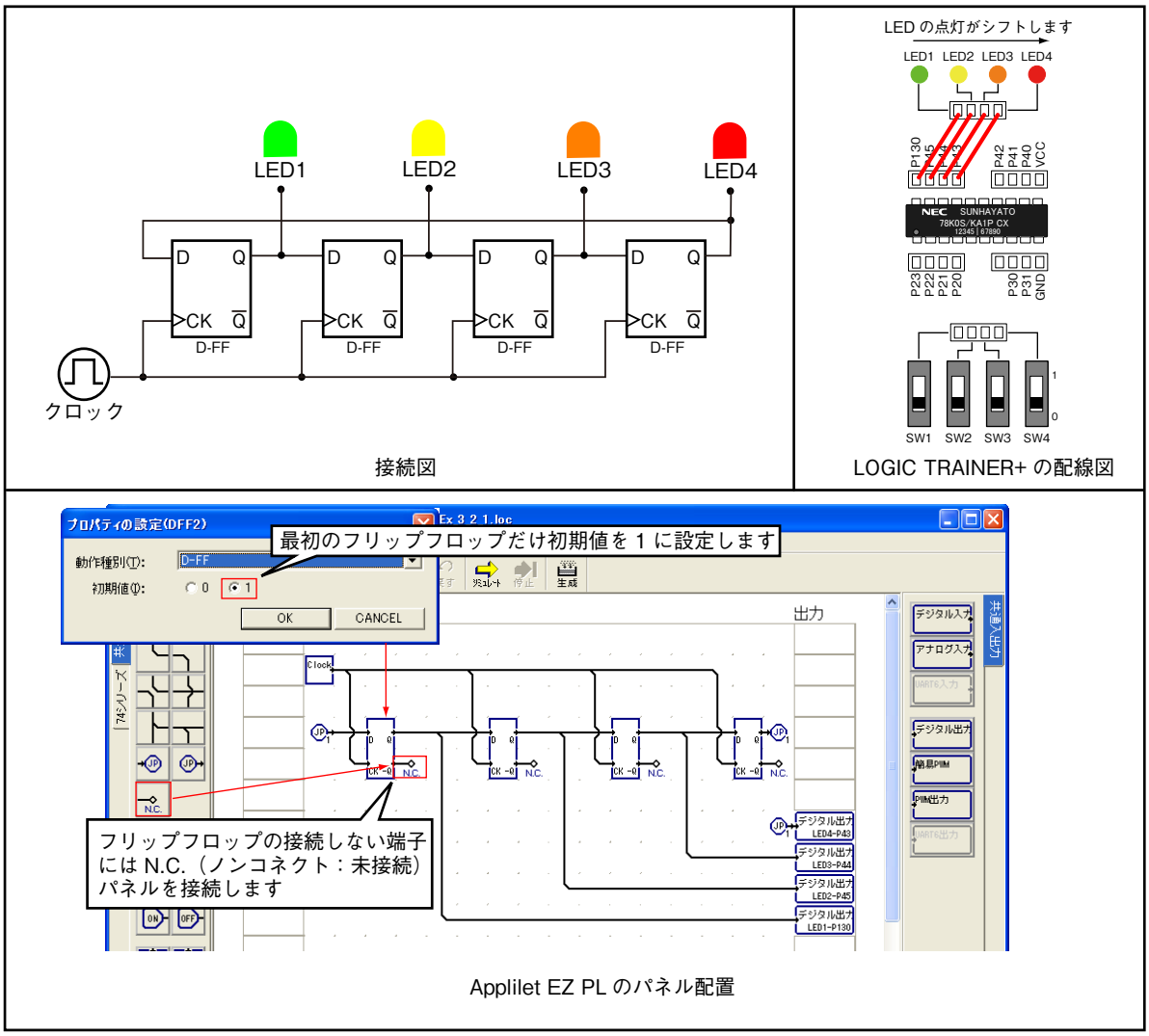


図3-16 Exercise 3.2.1

3.3 カウンタ

入力されたパルス信号のパルスの数を数える（カウントする）回路をカウンタといいます。パルスの立ち上がりで状態が変化するフリップフロップが使われます。コンピュータでは、短い周期のクロック信号をカウントして、より長い周期の信号をつくるタイマなどによく使われます。

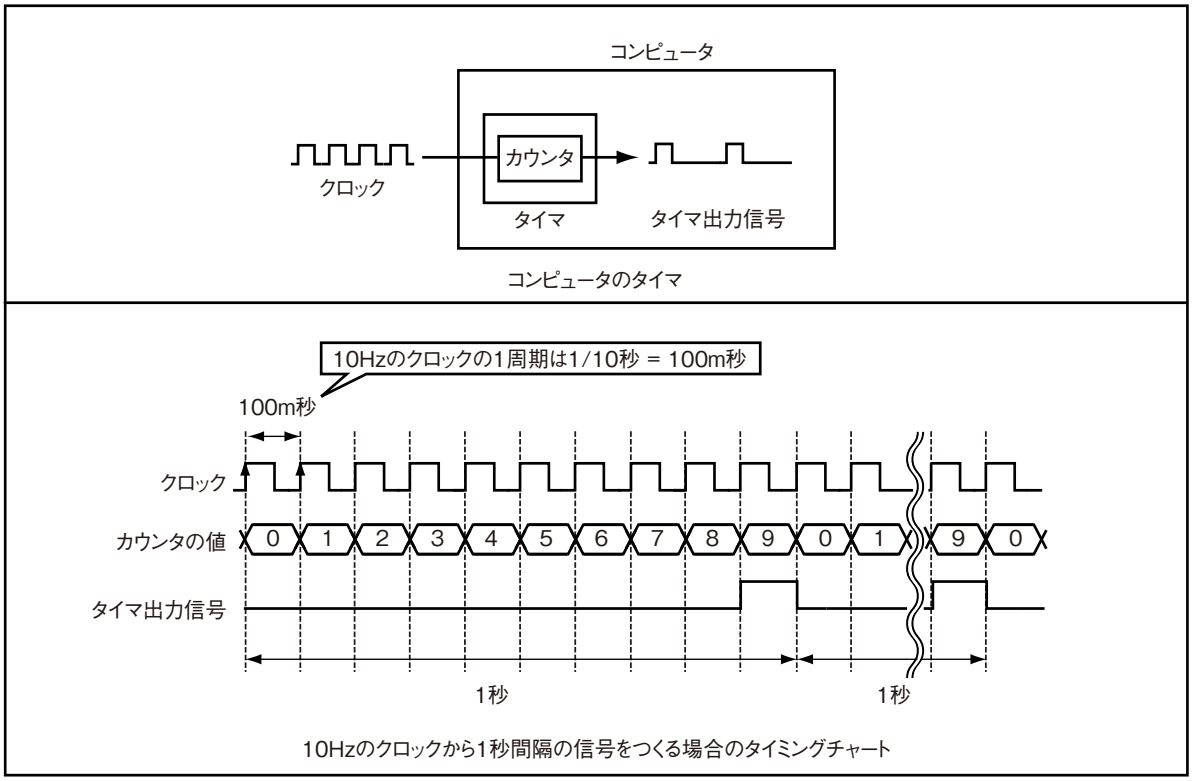


図3-17 コンピュータで使われるカウンタ（タイマ）

カウンタは実際には数える動作をしているというより、入力パルスの立ち上がり毎にフリップフロップの状態が変わり、その結果2進数でカウントしているようにみせかけています。

カウンタに使われるフリップフロップの数で、0からいくつまで数えられるかが決まります。例えばフリップフロップを2個使うと、2進数の00から11まで（10進数の0～3まで）カウントし、次は00に戻ります。これを4進カウンタと呼びます。n個のフリップフロップを使う場合、2^n進カウンタができることになります。

2進数について（3）

2進数の4桁を使って、10進数の1桁を表す方法をBCD（Binary Coded Decimal）コードといいます。

10進数とBCDコード

10進数	2進数
0	0000 0000
1	0000 0001
2	0000 0010
3	0000 0011
4	0000 0100
5	0000 0101
6	0000 0110
7	0000 0111
8	0000 1000
9	0000 1001
10	0001 0000
11	0001 0001

3.3.1 非同期式カウンタ

非同期式カウンタは、最前段のフリップフロップにのみパルスが入力され、あとのフリップフロップは前段のフリップフロップの状態の変化に応じて動作します。

非同期式カウンタの真理値表

パルス	Q2	Q1	$\overline{Q2}$	$\overline{Q1}$
0	0	0	1	1
1	1	1	0	0
2	1	0	0	1
3	0	1	1	0

←Q1、Q2 出力はダウンカウンタになります

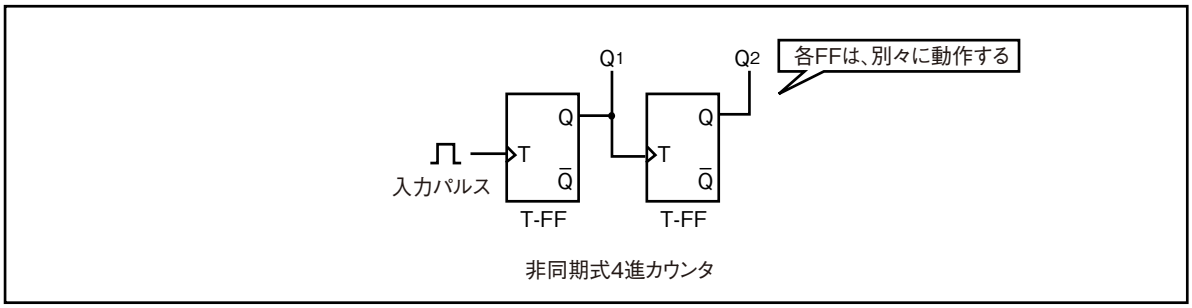


図3-18 非同期式カウンタ

非同期式カウンタは、各フリップフロップが無駄な動作をせず一見スマートですが、各フリップフロップの入力信号の変化から出力信号が変わるまでの時間（信号の伝播遅延時間といいます）が、後段につれて蓄積するため、注意が必要です。（Applilet EZ PL では非同期式カウンタでも伝播遅延時間は蓄積しません。）

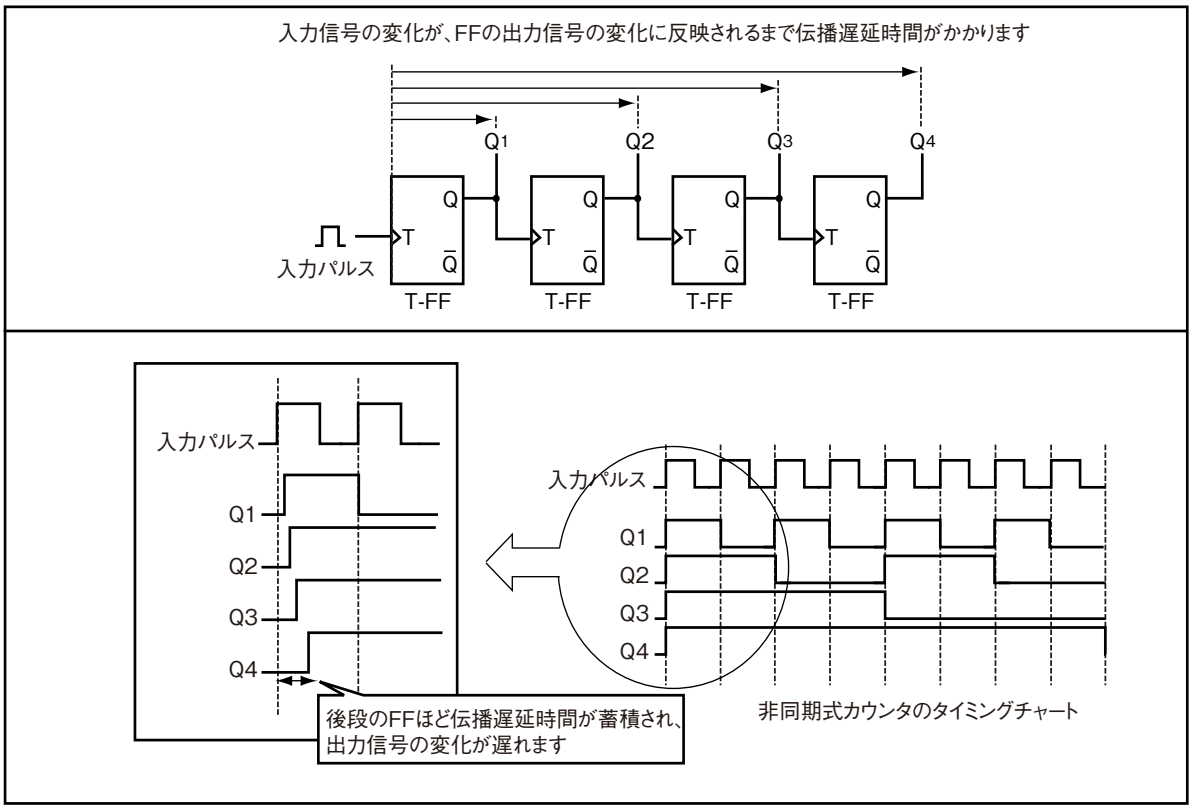


図3-19 非同期式カウンタの伝播遅延時間の蓄積

3.3.2 同期式カウンタ

同期式カウンタは、全てのフリップフロップに同じパルスが入力され、同じタイミングで動作します。非同期式に比べると、パルスの立ち上がり毎にかならず全てのフリップフロップが動作するので、消費電力は大きくなりますが、フリップフロップの遅延伝播時間は蓄積しないのでタイミングを考慮しやすく、大規模なものをつくる場合に設計しやすいというメリットがあります。

同期式カウンタの真理値表

パルス	Q2	Q1	$\overline{Q2}$	$\overline{Q1}$
0	0	0	1	1
1	0	1	1	0
2	1	0	0	1
3	1	1	0	0

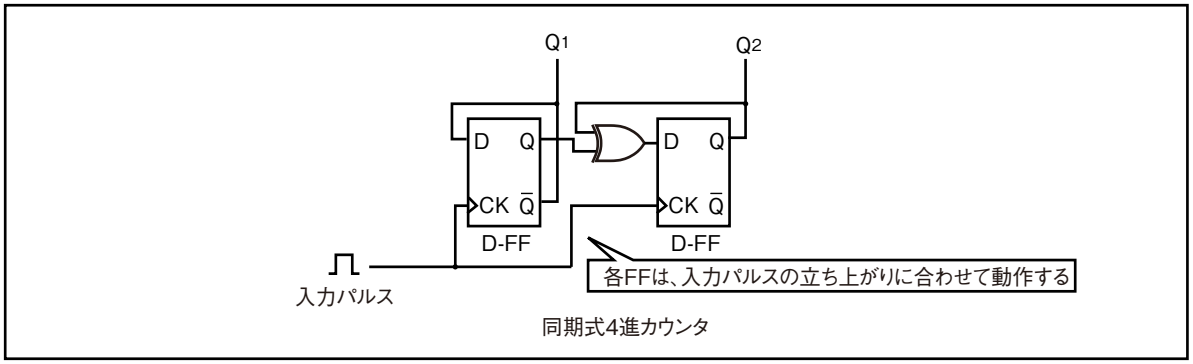


図3-20 同期式カウンタ

■ 同期式カウンタの考え方

2 ビットの同期式カウンタの遷移表（各フリップフロップの出力の現在と、次の状態を示した表です）から、考えます。現在の Q1、Q2 から次の Q1、Q2 になるような条件を、各フリップフロップの D 入力に入れることになります。

まず、Q1 は次の状態になるたびに反転しているので、自身の反転 $\overline{Q1}$ を D 入力にいれてやればよいことになります。

同期式カウンタの遷移表

現在		次	
Q2	Q1	Q2	Q1
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

← Q1 はいつも反転する

次に Q2 ですが、次に 1 になる条件をみると、現在の Q1、Q2 が以下の式のときであることがわかります。

$$D = (Q1 + \overline{Q2}) \cdot (\overline{Q1} + Q2)$$

同期式カウンタの遷移表

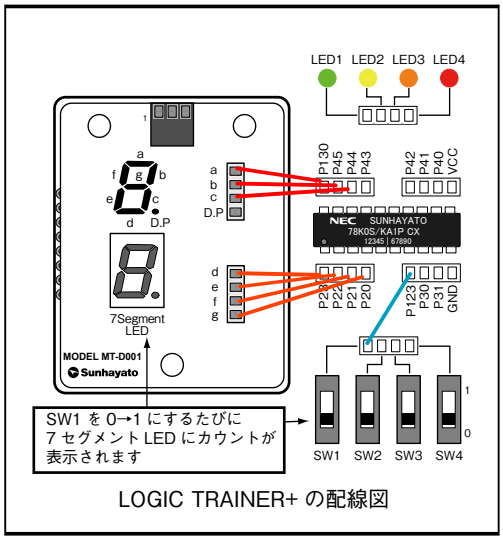
現在		次	
Q2	Q1	Q2	Q1
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

← Q2 が 1 になる条件

Exercise 3.3.①　非同期式、同期式カウンタをつくろう

非同期式カウンタと同期式カウンタの動作を LOGIC TRAINER+ で確認しましょう。パルスはスライドスイッチ SW1 から入力することとします。Applilet EZ PL シリーズの標準ロジック IC 74 シリーズパネルの 7 セグメントデコーダ回路（4511 パネル）を使って 7 セグメント LED に表示します。

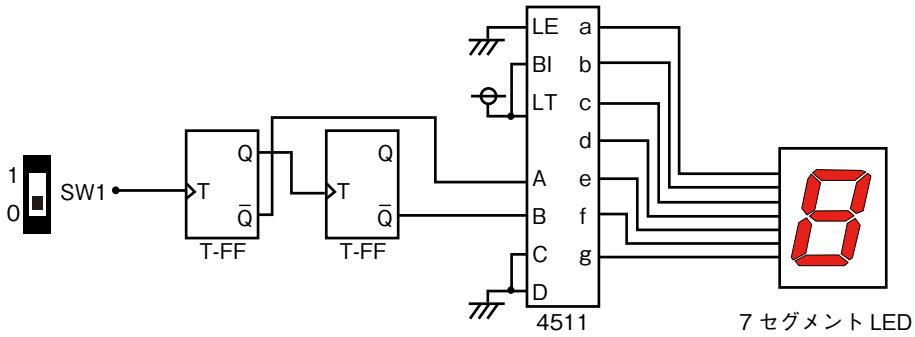
非同期式カウンタは、フリップフロップの出力 Q を接続するとダウンカウンタに、$\overline{Q}$ を接続するとアップカウンタになります。アップカウンタの場合は初期値「3」からカウントを開始することになります。



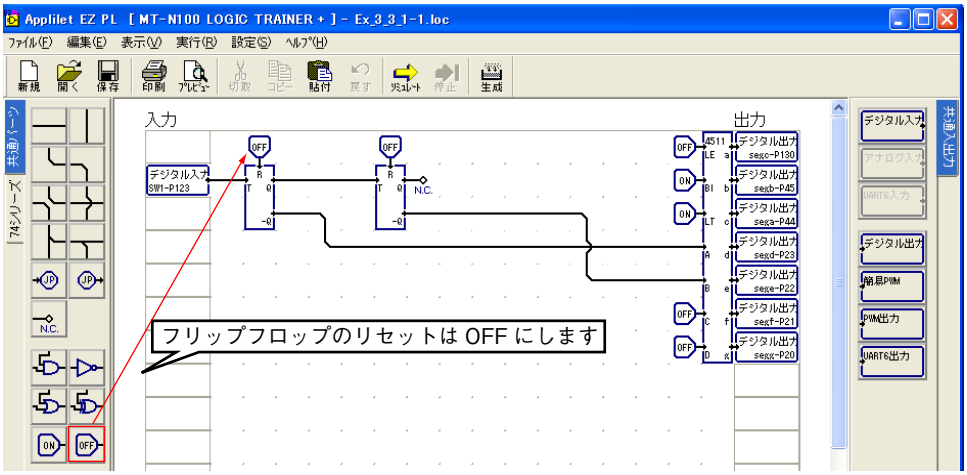
LOGIC TRAINER+ の配線図

7 セグメント LED デコーダ 4511 の入力信号

LE（Latch Enable）	＝1 入力で BCD コードをラッチする
BI（Blanking）	＝0 入力で全セグメントが消灯する
LT（Lamp Test）	＝0 入力で全セグメントが点灯する
A ～ D	＝4 桁の BCD コードを入力すると 7 セグメント LED に対応するパターンを表示する

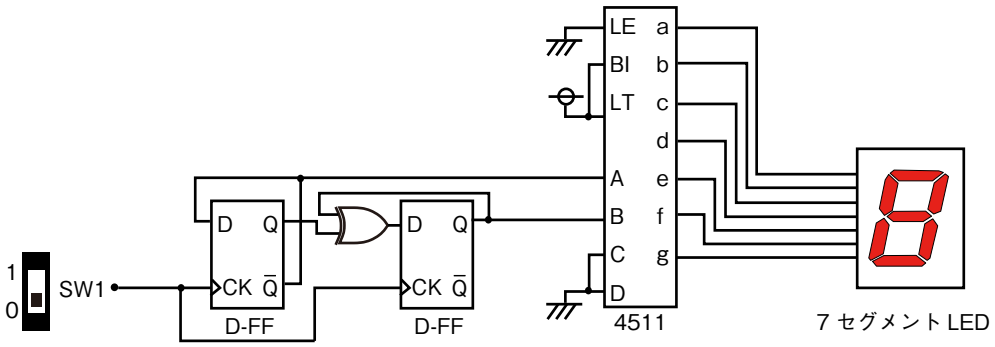


接続図（アップカウンタの場合）

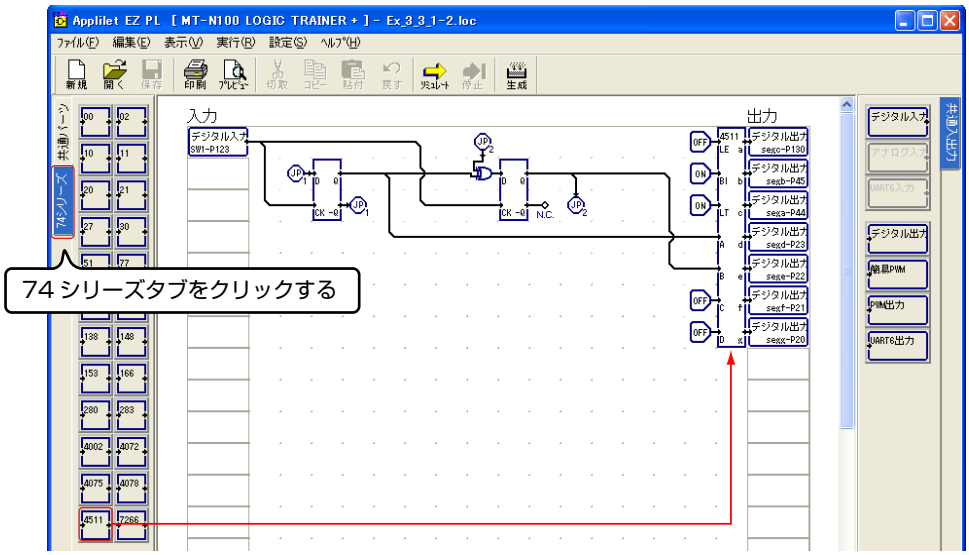


Applilet EZ PL のパネル配置

図3-21 LOGIC TRAINER+の配線図とExercise 3.3.①



接続図



Applilet EZ PL のパネル配置

図3-22 Exercise 3.3.②

標準ロジックICについて

標準ロジック IC とは、基本の論理ロジックからカウンタ、デコーダなど機能を持った論理回路を IC にパッケージしたものをいいます。最初にテキサス･インスツルメント社が出したシリーズが有名で、74xxx と 74 で始まる型番だったので、「74 シリーズ」と呼ばれます。Applilet EZ PL にも 74 シリーズのパネルが用意されています。

4. トラブル・シューティング

この章では、おもにLOGIC TRAINER+に論理回路を書込む際のトラブル･シューティングについて説明します。

4.1 回路が書込めない
4.2 仮想COM番号の確認方法

4.1 回路が書き込めない

4.1.1 評価ボードが回路書き込みの設定になっていない

プログラムを書き込む際は、LOGIC TAINER+ の MODE スイッチを「PROG」側へ設定してください。

4.1.2 Applilet EZ PLのCOMポート番号が合っていない

Applilet EZ PL で設定されている COM ポート番号と、開発用 PC で割り当てられている仮想 COM ポート番号が合っていない場合は、「4.2 仮想 COM ポート番号の確認方法」を参考に仮想 COM ポート番号を確認し、Applilet EZ PL に正しい COM ポート番号を設定してください。（Applilet EZ PL の「設定」→「フラッシュライタのオプション」メニューで設定します。）

4.2 仮想 COM ポート番号の確認方法

以下の方法で仮想 COM ポート番号を確認してください。

① Windows の「マイ コンピュータ」を右クリックし、「プロパティ」をクリックしてください。
② 「システムのプロパティ」ウィンドウの「ハードウェア」タブをクリックし、「デバイスマネージャ」をクリックします。
③ 一覧より「ポート（COM と LPT)」の「＋」マークをクリックすると、現在仮想 COM ポートドライバに割り当てられている COM ポートの番号が確認できます。

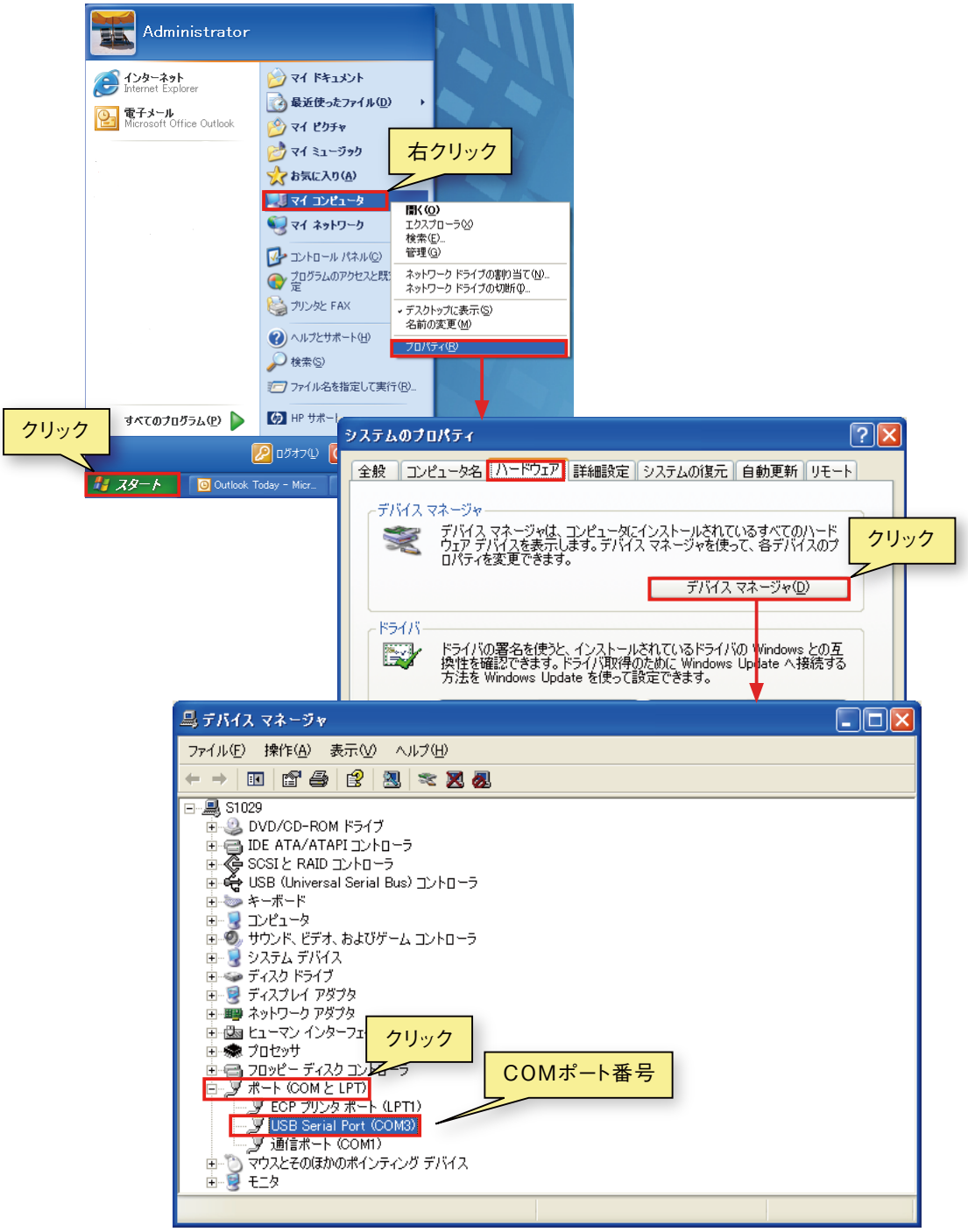


図4-1 仮想COMポート番号の確認

改訂履歴

Rev.	発行日	ページ	改訂内容
1.00	2008/04/20	-	初版発行

LOGIC TRAINER+
ガイドブック　ロジック基礎編

発行日　2008 年 4 月 20 日 Rev1.00
発　行　サンハヤト株式会社　〒 170-0005 東京都豊島区南大塚 3 丁目 40 番 1 号
©2008 Sunhayato Corp. All rights reserved.

SG043270